Harith Alani Lalana Kagal
Achille Fokoue Paul Groth
Chris Biemann Josiane Xavier Parreira
Lora Aroyo Natasha Noy
Chris Welty Krzysztof Janowicz (Eds.)

LNCS 8218

The Semantic Web – ISWC 2013

12th International Semantic Web Conference
Sydney, NSW, Australia, October 2013
Proceedings, Part I

Springer

Lecture Notes in Computer Science 8218

Commenced Publication in 1973
Founding and Former Series Editors:
Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Harith Alani Lalana Kagal
Achille Fokoue Paul Groth
Chris Biemann Josiane Xavier Parreira
Lora Aroyo Natasha Noy
Chris Welty Krzysztof Janowicz (Eds.)

The Semantic Web – ISWC 2013

12th International Semantic Web Conference
Sydney, NSW, Australia, October 21-25, 2013
Proceedings, Part I

Springer

Volume Editors

Harith Alani; The Open University, Milton Keynes, UK; h.alani@open.ac.uk

Lalana Kagal; Massachusetts Institute of Technology, USA; lkagal@csail.mit.edu

Achille Fokoue; IBM Research, Hawthorne, NY, USA; achille@us.ibm.com

Paul Groth; VU University Amsterdam, The Netherlands; p.t.groth@vu.nl

Chris Biemann; Technical University Darmstadt, Germany; biem@cs.tu-darmstadt.de

Josiane Xavier Parreira; DERI/NUIG, Galway, Ireland; josiane.parreira@deri.org

Lora Aroyo; VU University Amsterdam, The Netherlands; l.m.aroyo@cs.vu.nl

Natasha Noy; Stanford University, CA, USA; noy@stanford.edu

Chris Welty; IBM Research, Yorktown Heights, NY, USA; welty@us.ibm.com

Krzysztof Janowicz; University of California, Santa Barbara, USA; jano@geog.ucsb.edu

ISSN 0302-9743 e-ISSN 1611-3349
ISBN 978-3-642-41334-6 e-ISBN 978-3-642-41335-3
DOI 10.1007/978-3-642-41335-3
Springer Heidelberg New York Dordrecht London

Library of Congress Control Number: 2013949273

CR Subject Classification (1998): H.2, I.2, H.3, I.7, D.2, H.5, J.1

LNCS Sublibrary: SL 3 – Information Systems and Application, incl. Internet/Web and HCI

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Preface

How often have you seen a proceedings preface cited on Google Scholar? Or a person interviewed who refers to "that great preface article from ISWC-2013 that changed my life." A lot of work goes into organizing a conference and when it starts winding down towards the end and you suddenly realize, "Oh crap, I forgot to write the preface!" these thoughts of futility enter your mind and get in the way. I don't expect anyone will ever read this.

That said, I'm so honored to have the chance to introduce the proceedings for this, the twelfth in the ISWC series of conferences and proceedings. Twelve is, as I'm sure everyone reading this (which, as a mildly autistic logically-trained and obsessive-compulsive scientist, I have to point out is no one, making the statement tautologically true) knows, a very important number. It is divisible by 2,3,4, and 6, unlike the over-celebrated number 10, which is a stupid number that just so happens to correspond to the normal number of fingers. Why such an accidental freak of nature should cause this number to become so revered is beyond me. But I digress.

In a mere $2 \times 2 \times 3$ years, the Semantic Web has grown from a debateable and even controversial topic, discussed by a eclectic community of overly-enthusiastic entrepreneurs, web developers, and refugees from the AI winter, to an eclectic community of overly-enthusiastic entrepreneurs, web developers, and inductees into the AI Hall of Fame. Many of our articles are highly cited, and the conference itself enjoys a journal level impact factor.

I'm personally excited to express to all of you not reading this that the community has evolved significantly since the early days of the "machine-readable Web." Our proceedings reflects that evolution, with a large number of purely applied papers, some excellent data science, big data, and even a few papers on the dreaded "O" word. We seem to have moved past envisioning and designing the semantic web to using and experimenting with it – because without a doubt, the semantic web exists.

This volume contains the main proceedings of the International Semantic Web Conference (ISWC 2013), which was held in Sydney, Australia, in October 2013, making this the twelfth ISWC preface not to be read by anyone. The Research Track of the conference attracted 210 submissions, all of which *were* read (see below), and 45 of which were accepted, resulting in a 21% acceptance rate. The in-use track received 90 submissions and 16 papers were accepted, resulting in an 18% acceptance rate. Both the in-use and research tracks saw more submissions than last year and became more selective. Over the past ten years of unread ISWC prefaces, the research track submission numbers have fluctuated between 181 and 264, making this year typical. If we add in the evaluation and in-use tracks, overall submissions to the conference were 332, which is the second highest of all time.

Each paper received at least three, and sometimes as many as five, reviews from members of the Program Committee (impressive, indeed, compared to the number of readers of the preface). After the first round of reviews, authors had the opportunity to submit a rebuttal, leading to further discussions among the reviewers, a meta-review and a recommendation from a member of the Senior Program Committee (SPC). The SPC held a long virtual meeting in order to select the final set of accepted papers, paying special attention to papers that were borderline or had at least one recommendation for acceptance. In many cases, additional last- minute reviews were sought out to better inform the SPC's decision.

This edition of the International Semantic Web Conference marks the second year of the Evaluations and Experiments Track. The goal of this track is to consolidate research material and to gain new scientific insights and results by providing a place for in-depth experimental studies of significant scale. It aims at promoting experimental evaluations in Semantic Web/Linked Data domains where availability of experimental datasets and reproducibility of experiments are highly important. The Evaluations and Experiments track received 32 submissions from all areas of the Semantic Web. Ten papers were accepted, corresponding to a 31% acceptance rate. We consider this track to be in the incubator stage, and will continue to promote it in future years.

I cannot even begin to tell you, the non-reader of this preface, of the gratitude that we all owe to the excellent Organizing Committee, and especially to the local organizers Kerry Taylor and Armin Haller. Everyone worked incredibly hard to ensure the conference was a productive, informative and enjoyable experience, and receive nothing for their efforts beyond the satisfaction of seeing the conference go well, our hopefully-not-unexpressed gratitude, and having their names listed in the unread preface.

Chris Welty, General Chair on behalf of the editors:

August 2013

Harith Alani
Lalana Kagal
Research Track Chairs

Achille Fokoue
Paul Groth
In-Use Track Chairs

Chris Biemann
Josiane Xavier Parreira
Evaluation Track Chairs

Lora Aroyo
Natasha Noy
Doctoral Consortium Chairs

Krzysztof Janowicz
Proceedings Chair

Organization

Organizing Committee

General Chair

Chris Welty	IBM Research, USA

Vice Chair

Dimitrios Georgakopoulos	CSIRO, Australia

Local Chair

Kerry Taylor	CSIRO, Australia

Local Organisers

Armin Haller	CSIRO, Australia
Maxine Sherrin	Web Directions, Australia

Research Track Chairs

Harith Alani	Knowledge Media Institute, UK
Lalana Kagal	Massachusetts Institute of Technology, USA

In-Use Track Chairs

Achille Fokoue	IBM Research, USA
Paul Groth	VU University Amsterdam, The Netherlands

Evaluation Track Chairs

Chris Biemann	Technische Universität Darmstadt, Germany
Josiane Xavier Parreira	DERI/NUIG, Ireland

Doctoral Consortium Chairs

Lora Aroyo	VU University Amsterdam, The Netherlands
Natasha Noy	Stanford University, USA

Posters and Demos Chairs

Eva Blomqvist	Linköping University, Sweden
Tudor Groza	University of Queensland, Australia

Industry Track Chairs

Eric Franzon	WebMediaBrands, USA
Glenn Wightwick	IBM Research, Australia
Mary-Anne Williams	UTS, Australia

Workshops and Tutorials Chairs

Ben Johnston	UTS, Australia
Marta Sabou	MODUL University Vienna, Austria

Semantic Web Challenge Chairs

Sean Bechhofer	University of Manchester, UK
Andreas Harth	Karlsruhe Institute of Technology, Germany

Sponsorship Chairs

Pascal Hitzler	Wright State University, USA
Anni Rowland-Campbell	Intersticia, Australia

Publicity Chairs

Armin Haller	CSIRO, Australia
Yuan-Fang Li	Monash University, Australia
Kingsley Idehen	OpenLink Software, USA

Proceedings Chair

Krzysztof Janowicz	University of California, Santa Barbara, USA

Metadata Chairs

Li Ding	Memect, USA
Jie Bao	Samsung Information System America, USA

Student Coordinators

Raphaël Troncy	EURECOM, France
David Ratcliffe	CSIRO, Australia

Semantic Web Jam Session Chair

Aldo Gangemi	CNR Institute of Cognitive Sciences and Technology, Italy

Senior Program Committee – Research

Lora Aroyo	VU University Amsterdam, The Netherlands
Sören Auer	Universität Leipzig, Germany
Oscar Corcho	Universidad Politécnica de Madrid, Spain
Philippe Cudré-Mauroux	University of Fribourg, Switzerland
Tim Finin	University of Maryland, Baltimore County, USA
Fabien Gandon	INRIA, France
Asunción Gómez-Pérez	Universidad Politécnica de Madrid, Spain
Jeff Heflin	Lehigh University, USA
Martin Hepp	Bundeswehr University Munich, Germany

Pascal Hitzler	Kno.e.sis Center, Wright State University, USA
Andreas Hotho	University of Wuerzburg, Germany
David R. Karger	Massachusetts Institute of Technology, USA
Diana Maynard	University of Sheffield, UK
Dunja Mladenic	Jožef Stefan Institute, Slovenia
Jeff Z. Pan	University of Aberdeen, UK
Terry Payne	University of Liverpool, UK
Axel Polleres	Siemens AG Österreich, Austria
Steffen Staab	University of Koblenz-Landau, Germany

Program Committee – Research

Karl Aberer
Faisal Alkhateeb
Melliyal Annamalai
Kemafor Anyanwu
Knarig Arabshian
Manuel Atencia
Medha Atre
Jie Bao
Sean Bechhofer
Dominik Benz
Abraham Bernstein
Christian Bizer
Kalina Bontcheva
Paolo Bouquet
John Breslin
Christopher Brewster
Paul Buitelaar
Gregoire Burel
Diego Calvanese
Elizabeth Cano
Iván Cantador
Pierre-Antoine Champin
Gong Cheng
Smitashree Choudhury
Vassilis Christophides
Philipp Cimiano
Michael Compton
Gianluca Correndo
Isabel Cruz
Claudia D'Amato
Danica Damljanovic
Mathieu D'Aquin
Pieter De Leenheer
Mike Dean
Stefan Decker
Stefan Dietze
Li Ding
John Domingue
Michel Dumontier
Peter Eklund
Jérôme Euzenat
Anna Fensel
Miriam Fernandez
Achille Fokoue
Enrico Franconi
Bo Fu
Mark Gahegan
Aldo Gangemi
Raúl García-Castro
Nicholas Gibbins
Yolanda Gil
Fausto Giunchiglia
Birte Glimm
Jose Manuel Gomez-Perez
Olaf Görlitz
Alasdair J.G. Gray
Marko Grobelnik
Tudor Groza
Michael Gruninger
Christophe Guéret
Giancarlo Guizzardi
Armin Haller
Harry Halpin
Siegfried Handschuh
Lynda Hardman
Manfred Hauswirth

Sandro Hawke
Cory Henson
Rinke Hoekstra
Aidan Hogan
Laura Hollink
Matthew Horridge
Ian Horrocks
Katja Hose
Geert-Jan Houben
Bo Hu
Wei Hu
Eero Hyvönen
Krzysztof Janowicz
Mustafa Jarrar
Jason Jung
Hanmin Jung
Hong-Geen Kim
Matthias Klusch
Jacek Kopecky
Manolis Koubarakis
Matthias Knorr
Markus Krötzsch
Ora Lassila
Jens Lehmann
Freddy Lecue
Juanzi Li
Yuefeng Li
Vanessa Lopez
Frederick Maier
Deborah McGuinness
Peter Mika
Alessandra Mileo
Riichiro Mizoguchi
Luc Moreau
Boris Motik
Enrico Motta
Mark Musen
Ekawit Nantajeewarawat
Nadeschda Nikitina
Andriy Nikolov
Natasha F. Noy
Kieron O'Hara
Massimo Paolucci
Bijan Parsia
Alexandre Passant
Carlos Pedrinaci
Sofia Pinto
Dimitris Plexousakis
Valentina Presutti
Abir Qasem
Guilin Qi
Riccardo Rosati
Matthew Rowe
Sebastian Rudolph
Marta Sabou
Harald Sack
Hassan Saif
Manuel Salvadores
Ulrike Sattler
Luigi Sauro
Francois Scharffe
Ansgar Scherp
Stefan Schlobach
Daniel Schwabe
Juan F. Sequeda
Luciano Serafini
Milan Stankovic
Umberto Straccia
Markus Strohmaier
Rudi Studer
Gerd Stumme
Jing Sun
Hideaki Takeda
Valentina Tamma
Kerry Taylor
Krishnaprasad Thirunarayan
Raphaël Troncy
Tania Tudorache
Giovanni Tummarello
Anni-Yasmin Turhan
Victoria Uren
Maria Esther Vidal
Tomas Vitvar
Johanna Völker
Claudia Wagner
Haofen Wang
Zhichun Wang
Kewen Wang
Fang Wei-Kleiner
Fouad Zablith
Antoine Zimmermann

Additional Reviewers – Research

Alessandro Adamou
Nitish Aggarwal
Zaenal Akbar
Mustafa Al-Bakri
Pramod Anantharam
Mihael Arcan
Ana Armas
Alessandro Artale
Samantha Bail
Kallola Bal
Cosmin Basca
David Berry
Nicola Bertolin
Daniel Borchmann
Georgeta Bordea
Stefano Botoli
Janez Brank
Lorenz Bühmann
Jean-Paul Calbimonte
Delroy Cameron
Stephane Campinas
Iván Cantador
Xiaoqi Cao
David Carral
Olivier Corby
Luca Costabello
Philippe Cudré-Mauroux
Olivier Curé
Maciej Dabrowski
Evangelia Daskalaki
Steven de Rooij
Christophe Debruyne
Renaud Delbru
Gianluca Demartini
Leon Derczynski
Laura Dragan
Timofey Ermilov
Nicola Fanizzi
Catherine Faron Zucker
Mariano Fernández-López
Alfio Ferrara
Daniel Fleischhacker
Giorgos Flouris
Andre Freitas
Sarah Alice Gaggl
Venkat Raghavan Ganesh Sekar
Daniel Garijo
Rafael S. Goncalves
Thomas Gottron
Jorge Gracia
Gerd Gröner
Kalpa Gunaratna
Tian Guo
Masahiro Hamasaki
Rakebul Hasan
Yulan He
Katja Hose
Lei Hou
Myunggwon Hwang
Ali Intizar
Ernesto Jimenez-Ruiz
Sung-Jae Jung
Martin Junghans
Ken Kaneiwa
Patrick Kapahnke
Hyeongsik Kim
Sabrina Kirrane
Szymon Klarman
Christoph Kling
Johannes Knopp
Matthias Knorr
Magnus Knuth
Ilianna Kollia
Patrick Koopmann
Harshit Kumar
Jérôme Kunegis
Sarasi Lalithsena
Sungin Lee
Zhixing Li
Marcel Lippmann
Nuno Lopes
Esther Lozano
Frederick Maier
Albert Merono-Penuela
Patrick Minder
Pasquale Minervini

Raghava Mutharaju
Yavor Nenov
Matthias Nickles
Vit Novacek
Andrea Giovanni Nuzzolese
Emmanuel Pietriga
Robert Piro
Denis Ponomaryov
Behrang Qasemizadeh
Nguyen Quoc Viet Hung
David Ratcliffe
Yuan Ren
Achim Rettinger
Mehdi Riahi
Bene Rodriguez
Michael Rogger
Cristina Sarasua
Luigi Sauro
Thomas Scharrenbach
Stefan Scheglmann
Oshani Seneviratne
Chao Shao
Philipp Singer
Dezhao Song
Claus Stadler
Ioannis Stavrakantonakis
Kostas Stefanidis
Giorgio Stefanoni
Andreas Steigmiller
Nadine Steinmetz
Mari Carmen Suárez-Figueroa
Fabio Tacchelli
Jiao Tao
Veronika Thost
Aibo Tian
Konstantin Todorov
Trung-Kien Tran
Dmitry Tsarkov
Petros Tsialiamanis
Sujan Udayanga
Jung-Ho Um
Jürgen Umbrich
Joerg Unbehauen
Matteo Vasirani
Mihaela Verman
Daniel Vila-Suero
Serena Villata
Christian von der Weth
Joerg Waitelonis
Simon Walk
Zhe Wang
Zhigang Wang
Guohui Xiao
Xiaowang Zhang
Yuting Zhao
Dmitriy Zheleznyakov
Yujiao Zhou

Program Committee – Semantic Web In-Use

Dean Allemang
Anupriya Ankolekar
Phil Archer
Christian Bizer
Jerven Bolleman
Gully Burns
Iván Cantador
Vinay Chaudhri
Michelle Cheatham
Paolo Ciccarese
Oscar Corcho
Gianluca Correndo
Mathieu D'Aquin
Brian Davis
Mike Dean
Ying Ding
Leigh Dodds
Michel Dumontier
Federico Michele Facca
Achille Fokoue
Alasdair Gray
Paul Groth
Tudor Groza
Peter Haase

Armin Haller
Siegfried Handschuh
Lee Harland
Steve Harris
Martin Hepp
Ivan Herman
Matthew Horridge
Wei Hu
Prateek Jain
Krzysztof Janowicz
Pavel Klinov
Matthias Klusch
Spyros Kotoulas
Christoph Lange
Yuan-Fang Li
Thorsten Liebig
Antonis Loizou
Akshay Maan
Pablo Mendes
Lyndon Nixon
Massimo Paolucci
Alexandre Passant
Carlos Pedrinaci
Edoardo Pignotti
Axel Polleres
Héctor Pérez-Urbina
Yves Raimond
Cartic Ramakrishnan
Marco Rospocher
Matthew Rowe
Marta Sabou
Manuel Salvadores
Marc Schaaf
Michael Schmidt
Juan F. Sequeda
Milan Stankovic
Nenad Stojanovic
Tania Tudorache
Mischa Tuffield
Giovanni Tummarello
Michael Uschold
Willem Robert Van Hage
Jacco Van Ossenbruggen
Ruben Verborgh
Holger Wache
Jesse Jiaxin Wang
Kewen Wang
Egon Willighagen
Zhe Wu
Fouad Zablith
Amapali Zaveri
Amal Zouaq

Additional Reviewers – Semantic Web In-Use

Maribel Acosta
Jay Banerjee
Victor de Boer
Claudio Giovanoli
Vit Novacek
Fabrizio Orlandi
Behrang Qasemizadeh
Padmashree Ravindra
Laurens Rietveld
Bene Rodriguez-Castro
Edgar Rodriguez-Diaz
Anisa Rula
Simon Scerri
Stefan Schlobach
Alex Stolz
Jiao Tao
Läszló Török
Zhe Wang

Senior Program Committee – Evaluations and Experiments

Sören Auer	University of Leipzig, Germany
Abraham Bernstein	University of Zurich, Switzerland
Philipp Cimiano	University of Bielefeld, Germany
Jérôme Euzenat	INRIA, France
Manfred Hauswirth	DERI/NUI Galway, Ireland
Heiner Stuckenschmidt	University of Mannheim, Germany

Program Committee – Evaluations and Experiments

Denilson Barbosa
Mihaela Bornea
Carlos Castillo
Christian Chiarcos
Vassilis Christophides
Oscar Corcho
Ernesto William De Luca
Raúl García-Castro
Andreas Harth
Gregor Heinrich
Sebastian Hellmann
Robert Hoehndorf
Aidan Hogan
Ryutaro Ichise
Antoine Isaac
Ernesto Jimenez-Ruiz
Manolis Koubarakis
Thomas Krennwallner
Jose Emilio Labra Gayo
Diana Maynard
Christian Meilicke
Axel-Cyrille Ngonga Ngomo
Heiko Paulheim
Axel Polleres
Simone Paolo Ponzetto
Marta Sabou
Sherif Sakr
Kai-Uwe Sattler
Fabian M. Suchanek
Ondrej Svab-Zamazal
Pierre-Yves Vandenbussche
Maria Esther Vidal
Shenghui Wang
René Witte
Stuart Wrigley

Program Committee – Doctoral Consortium

Abraham Bernstein
Oscar Corcho
Mathieu D'Aquin
Enrico Motta
Marta Sabou
Guus Schreiber
Elena Simperl
David Karger
Diana Maynard

Sponsors

Student Travel Award Sponsor
Semantic Web Science Association (SWSA)
National Science Foundation (NSF)

Invited Speakers Sponsor
Artificial Intelligence Journal

Semantic Web Challenge Sponsor
Elsevier

Platinum
Bing™
fluid Operations
Microsoft Research™
PreviousNext

Gold
IBM Research
Ontotext
Yahoo!

Silver
IOS Press
OpenLink Software

Local Organizers
CSIRO
W3C Australia
Web Directions South

Keynote Talks
(Abstracts)

Progress in Open-World, Integrative, Transparent, Collaborative Science Data Platforms

Peter Fox

Tetherless World Constellation
Rensselaer Polytechnic Institute, US
pfox@cs.rpi.edu

Abstract

As collaborative, or network science spreads into more science, engineering and medical fields, both the participants and their funders have expressed a very strong desire for highly functional data and information capabilities that are a) easy to use, b) integrated in a variety of ways, c) leverage prior investments and keep pace with rapid technical change, and d) are not expensive or time-consuming to build or maintain. In response, and based on our accummulated experience over the last decade and a maturing of several key semantic web approaches, we have adapted, extended, and integrated several open source applications and frameworks that handle major portions of functionality for these platforms. At minimum, these functions include: an object-type repository, collaboration tools, an ability to identify and manage all key entities in the platform, and an integrated portal to manage diverse content and applications, with varied access levels and privacy options.

At the same time, there is increasing attention to how researchers present and explain results based on interpretation of increasingly diverse and heterogeneous data and information sources. With the renewed emphasis on good data practices, informatics practitioners have responded to this challenge with maturing informatics-based approaches. These approaches include, but are not limited to, use case development; information modeling and architectures; elaborating vocabularies; mediating interfaces to data and related services on the Web; and traceable provenance. The current era of data-intensive research presents numerous challenges to both individuals and research teams. In environmental science especially, sub-fields that were data-poor are becoming data-rich (volume, type and mode), while some that were largely model/ simulation driven are now dramatically shifting to data-driven or least to data-model assimilation approaches. These paradigm shifts make it very hard for researchers used to one mode to shift to another, let alone produce products of their work that are usable or understandable by non-specialists. However, it is exactly at these frontiers where much of the exciting environmental science needs to be performed and appreciated.

Research networks (even small ones) need to deal with people, and many intellectual artifacts produced or consumed in research, organizational and/our outreach activities, as well as the relations among them. Increasingly these networks are modeled as knowledge networks, i.e. graphs with named and typed relations among the 'nodes'. Some important nodes are: people, organizations, datasets, events, presentations, publications, videos, meetings, reports, groups, and more. In this heterogeneous ecosystem, it is important to use a set of common informatics approaches to co-design and co-evolve the needed science data platforms based on what real people want to use them for.

We present our methods and results for information modeling, adapting, integrating and evolving a networked data science and information architecture based on several open source technologies (e.g. Drupal, VIVO, the Comprehensive Knowledge Archive Network; CKAN, and the Global Handle System; GHS) and many semantic technologies. We discuss the results in the context of the Deep Carbon Virtual Observatory and the Global Change Information System, and conclude with musings on how the smart mediation among the components is modeled and managed, and its general applicability and efficacy.

Light at the End of the Tunnel

Ramanathan V. Guha

Google Inc., US
guha@guha.com

Abstract

A significant fraction of the pages on the web are generated from structured databases. A longstanding goal of the semantic web initiative is to get webmasters to make this structured data directly available on the web. The path towards this objective has been rocky at best. While there have been some notable wins (such as RSS and FOAF), many of the other initiatives have seen little industry adoption. Learning from these earlier attempts has guided the development of schema.org, which appears to have altered the trajectory. Two years after its launch over 4 million Internet domains are are using schema.org markup.

In this talk, we recount the history behind the early efforts and try to understand why some of them succeeded while others failed. We will then give an update on Schema.org, its goals, accomplishments and where it is headed. We will also discuss some of the interesting research problems being addressed in the context of this effort.

Semantic Big Data in Australia – From Dingoes to Drysdale

Jane Hunter

School of ITEE, The University of Queensland, Australia
j.hunter@uq.edu.au

Abstract

This keynote will describe a number of projects being undertaken at the University of Queensland eResearch Lab that are pushing Semantic Web technologies to their limit to help solve grand challenges in the environmental, cultural and medical domains. In each of these use cases, we are integrating multi-modal data streams across space, time, disciplines, formats and agencies to infer and expose new knowledge through rich multi-layered and interactive visualizations. We are developing hypothesis-based query interfaces that provide evidence to validate or refute hypotheses and decision support services that recommend the optimum actions given current or predicted scenarios. We are using ontologies to influence and adapt government policies by linking policy-driven implementations, investments and management actions to real world indicators. Through evaluation of the methods and assessment of the achievements associated with the OzTrack [1,2], eReef [3], Skeletome[4] and Twentieth Century in Paint[5] projects, I will highlight those Semantic Web technologies that have worked for us and our user communities, those that haven't and those that need improvement. Finally I will discuss what I believe will be the major outstanding research challenges facing Semantic Big Data in the next 5 years and those research areas with the greatest potential for impact.

References

1. J.Hunter, C.Brooking, W.Brimblecombe, R.Dwyer, H.Campbell, M.Watts, C.Franklin, OzTrack – e-Infrastructure to support the Management, Analysis and Sharing of Animal Tracking Data, IEEE eScience, Beijing, October 2013.
2. L.Gao, H. Campbell, O. Bidder and J. Hunter, A Web-based Semantic Tagging and Activity Recognition System for Species' Accelerometry Data, Elsevier, Ecological Informatics, Vol 13, January 2012, pp 47-56.
3. J.Hunter, A.Gebers, T.Dettrick, Automating Online Reef Report Cards – Linking Land Management Practices to Water Quality and Coral Reef Ecosystem Health, HAICTA 2013, September, 2013, Corfu, Greece.

4. T. Groza, A. Zankl, Y-F Li, J.Hunter, Using Semantic Web Technologies to Build a Community-driven Knowledge Curation Platform for the Skeletal Dysplasia Domain, ISWC 2011 In Use Track, 2011.
5. J. Hunter and S. Odat, Building a Semantic Knowledge-Base for Painting Conservators, IEEE eScience 2011, Stockholm, Dec 6-8, 2011.

Table of Contents – Part I

Research Track

Table of Contents – Part II

In-Use Track

Evaluations and Experiments Track

Doctoral Consortium – Selected Papers

TRM – Learning Dependencies between Text and Structure with Topical Relational Models

Veli Bicer[1], Thanh Tran[2], Yongtao Ma[2], and Rudi Studer[2]

[1] IBM Research, Smarter Cities Technology Centre
Damastown Industrial Estate, Dublin, Ireland
velibice@ie.ibm.com
[2] Institute AIFB, Karlsruhe Institute of Technology
Geb. 11.40, KIT-Campus Süd, Karlsruhe, Germany
{duc.tran,yongtao.ma,rudi.studer}@kit.edu

Abstract. Text-rich structured data become more and more ubiquitous on the Web and on the enterprise databases by encoding heterogeneous structural information between entities such as people, locations, or organizations and the associated textual information. For analyzing this type of data, existing topic modeling approaches, which are highly tailored toward document collections, require manually-defined regularization terms to exploit and to bias the topic learning towards structure information. We propose an approach, called *Topical Relational Model*, as a principled approach for automatically learning topics from both textual and structure information. Using a topic model, we can show that our approach is effective in exploiting heterogeneous structure information, outperforming a state-of-the-art approach that requires manually-tuned regularization.

1 Introduction

We study the problem of learning on *text-rich structured data* that shares these main characteristics: it describes interconnected objects (*relational*) of different types (*heterogeneous*) that are associated with textual attributes (*text-rich*). Examples include graph-structured RDF data forming connected resource descriptions and relational database records containing textual values that are connected via foreign keys.

Topic modeling (TM) approaches have shown to be effective for dealing with text, which recently, have also been extended to deal with the combination of textual and structured data [1–9]. However, when dealing with the text-rich structured data as we consider as an input in this paper (as shown Fig. 1), two problems mainly arise: First, such data mostly consists of a heterogeneous structure such as many classes and relations each of which has varying effects on different topics. Thus entities having different structure information in the data need to have different topic distributions. For example, any `Company` entity having `product` relation is more related to a topic about *manufacturing* than another `Company` entity of being a movie distributor. Usually such correlations

H. Alani et al. (Eds.): ISWC 2013, Part I, LNCS 8218, pp. 1–16, 2013.

between the topics and these structural elements (i.e. classes and relations) are not one-to-one, but the latter can be correlated to one or more topics with different proportions. Previous TM approaches are not well suited to handle such complex correlations between the structure and text since they either consider a homogeneous structure (e.g. social networks [5], citation networks [7] or Web links [8]) or networks with a few types of relations [9, 7]. An alternative solution to handle such complex correlations can also be applying *Statistical Relational Learning* (SRL) techniques [10] which are naturally formulated as instances of learning a probabilistic model (e.g. Markov Logic Network (MLN) [11] or (non-)linear link functions [12]) from the data. However the main problem of directly applying the SRL techniques into our setting is about handling *textual data* since they consider the input data to be available in a sort of structured form (e.g. a user-movie matrix indicating whether a user likes a movie). Only few works exist that utilize SRL to learn from text-rich structured data [13, 12] but mainly suffer from a large and complex structured network required for textual data. At this point, using the latent topics as low-dimensional representation of textual data provides a better way to incorporate textual information into the SRL models.

Yet another problem also arises due to *the sparsity of these correlations* between the topics and the structure. For example, in Fig. 2 the class `City` is highly correlated to the topic t_4 but not to the other three topics. Such sparsity is not well addressed by the previous work, especially the ones focusing on heterogeneous networks (e.g. [9, 7]) which aims the topic smoothness instead of sparsity. One exception to this is the focused topic model [14] which utilizes *latent feature models* to control sparsity but it is an unsupervised approach and does not take the structure information into account.

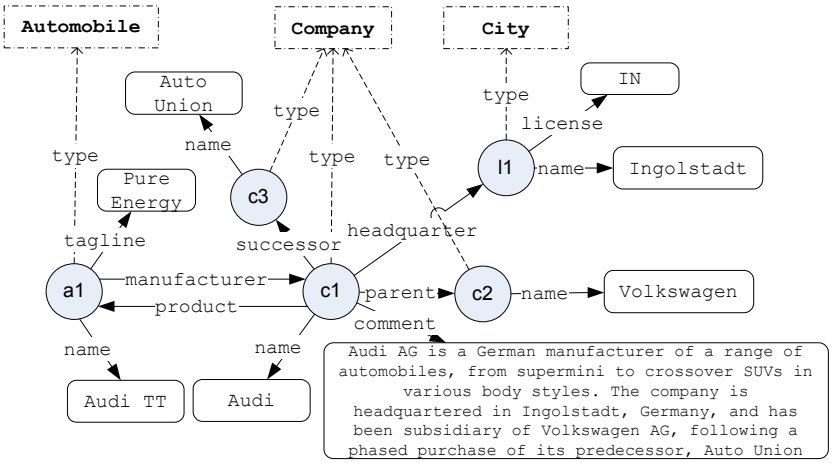


Fig. 1. An example data graph

In this paper, we propose *Topical Relational Model* (TRM) that uses relational information in structured data for topic modeling (text analysis tasks), and also allows the learned topics to be employed as a low-dimensional representation of the text to capture dependencies between structured data objects. The main novel aspects of this model are: (1) compared to previous SRL works, TRM employs hidden topic variables to reduce model complexity. TRM uses latent topics to represent textual information, targeting the specific case of text-rich structured data. In this way, it *provides a systematic way to learn low-dimensional features from text* (i.e., topics) that can be used for SRL-related tasks. We show in experiments that topics learned by TRM outperform the features manually specified in previous SRL works [12, 11]. (2) Compared to existing TM approaches, TRM is able to exploit the richness in relational information available in structured data. With TRM, the learning of topics recognizes the heterogeneity of classes and relations associated with entities. While some related TM work [9] requires manually defined regularization terms to exploit this heterogeneous structure information, TRM captures correlations between structure and topics through dedicated latent feature model parameter in order to better handle the sparsity of the correlations. In experiments, we show that leveraging relational information this way, TRM improves the performance of state-of-the-art TM approaches [9, 15]. (3) TRM is unique in its hybrid nature: it is a *topic model* that incorporates structure in addition to textual information; at the same time, it is also a *Bayesian network capturing relational dependencies* between text-rich objects that can be employed for SRL tasks.

Structure. We present the main ideas behind TRM in Sec. 2.1, TRM variables in Sec. 2.2 and their dependencies in Sec. 2.3. Sec. 2.4 describes the generative process, which is reversed in Sec. 2.5 for learning TRM. Experimental results are presented in Sec. 3, followed by conclusions in Sec. 4.

2 Topical Relational Models

TRM supports different types of *graph-structured data* including relational, XML and RDF data. The focus lies on text-rich data describing objects through textual attributes. More formally, the data is a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{R})$ (see Fig.1), where $\mathcal{V}$ is the disjoint union $\mathcal{V} = \mathcal{V}_C \uplus \mathcal{V}_E$ representing *classes* and *entities*, respectively, and $\mathcal{R}$ stands for binary *relations* between entities. The set of classes an entity e belongs to is denoted $C(e) = \{c \mid type(e, c)\}$ (*type* is a special relation that connects an entity node with a class node), while the relations e is involved in is $R(e) = \{r \mid r(e, e'), r(e', e) \in \mathcal{R} \wedge e, e' \in \mathcal{V}_E\}$. In our text-rich data setting, every entity also has some textual attribute values such as `name`, `comment` and `description`. To incorporate this, every entity node $e \in \mathcal{V}_E$ is treated as a document, i.e., modeled as a bag of words $e = \{w_1, \ldots, w_{|e|}\}$, which contains all words in the textual values of e.

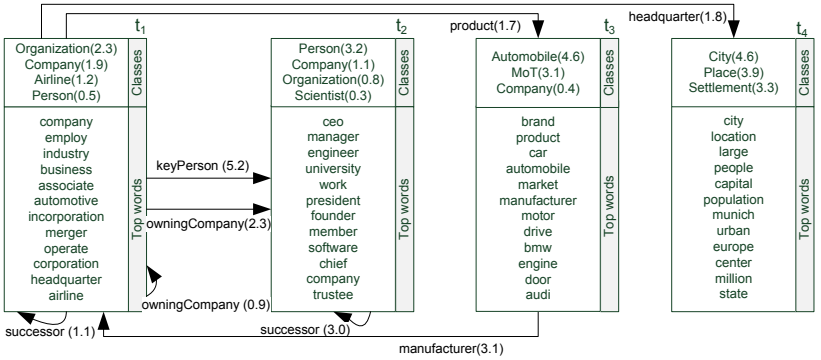


Fig. 2. An excerpt of the TRM result for DBpedia. It captures four TRM topics and and their top ranked words. Further, classes that are highly correlated with individual topics and relations that are highly correlated with pairs of topics are shown (strength of correlation shown in brackets).

2.1 TRM

TRM is a *topic model* representing $\mathcal{G}$ through a set of topics $T = \{t_i, \ldots, t_K\}$. Each $t \in T$ is a probabilistic distribution $\{p(w \mid t)\}_{w \in V}$, where $\sum_{w \in V} p(w \mid t) = 1$ and V is the vocabulary of words. However, the context from which topics are derived is not made up of plain words but also includes entities, classes and relations in $\mathcal{G}$. This is reflected in the TRM's output, which includes topics as well as their strength of correlation w.r.t. words, classes and relations. Fig. 2 illustrates this. For instance, we can see that the related words `employ` and `merger` form the topic t_1. Further, t_1 is not only drawn from words but also, correlates with entities of the types `Organization` and `Company`. This topic (and its associated words) correlates with other topics (words) such as t_3 (`brand` and `engine`) in the context of the `manufacturer` relation.

Because TRM also captures dependencies between structure elements (classes and relations), it can also be seen as a *SRL approach*. However, it captures them only indirectly using topics, which act as a low-dimensional abstraction of the text. We now present the TRM's Bayesian network representation to show these relational dependencies conditioned on topics.

2.2 Template-Based Representation of TRM

A template-based representation of Bayesian networks [16] is used to define TRM as a set of *template attributes* and *template factors*.

Let $\mathbf{X}$ denotes some of the random variables, $\mathbf{X} = \{X_1, ..., X_n\}$, where each $X_i \in \mathbf{X}$ can be assigned a value from the range $Val(X_i)$. Then, a *template attribute* (or attribute hereafter) is a function $A(\alpha_1, ..., \alpha_k)$, whose range is $Val(A)$, and each argument α_i is a placeholder to be instantiated with values of a particular type. For example, there are the template attributes $Company(\alpha_1)$ and $headquarter(\alpha_1, \alpha_2)$. The values used to instantiate α_i are drawn from the data,

called the *object skeleton*, $\mathcal{O}(A)$. Given $\mathcal{O}(A)$, the variables instantiating A is $\mathbf{X}_{\mathcal{O}(A)} = \{A(o) \mid o \in \mathcal{O}(A)\}$, where $Val(X_i) = Val(A)$ for each $X_i \in \mathbf{X}_{\mathcal{O}(A)}$. For example, from $Company(\alpha_1)$ and $\mathcal{O}(Company) = \{c1, c2, c3\}$, we obtain the random variables $\mathbf{X}_{\mathcal{O}(Company)} = \{Company(c1), Company(c2), Company(c3)\}$.

Template factors are used to define probability distributions over random variables. They are templates because instead of ground variables, template attributes are taken as arguments. They return real numbers for assignments of variables instantiated from template attributes, i.e., given a tuple of attributes $A_1, ..., A_l$, a template factor f is a function from $Val(A_1) \times ... \times Val(A_l)$ to $\mathbb{R}$. A special template factor is the *conditional probability distribution*, which splits the attributes into two groups, $\mathbf{A}_c, \mathbf{A}_{Pa} \subseteq \{A_1, ..., A_l\}$, called child and parent attributes.

Observed Variables. Elements in $\mathcal{G}$ are used to instantiate three template attributes defined for observed variables, namely the *class* c, the *relation* r and the *entity-word assignment* w. Their object skeletons are entities belonging to the type c, relations of the type r, and words in the entities' bag-of-words representation, i.e., $\mathcal{O}(c) = \{e|c \in C(e)\}$, $\mathcal{O}(r) = \{(e, e')|r(e, e') \in \mathcal{R}\}$ and $\mathcal{O}(w) = \{(e, v) \mid e \in \mathcal{V}_E \wedge v \in e\}$. These templates are binary-valued functions, indicating whether an entity, a relation instance or an entity-word assignment exists or not. For example, the variables $Company(c1)$, $product(c1, p1)$ and $w(c1, car)$ obtained for these templates model whether there is an entity `c1` that is a company, `p1` is a product of `c1` and `car` is a word associated with `c1`, respectively.

Observe that some entity-word assignments are dependent on a particular relation. For instance, the probability of observing the assignment $w(e, munich)$ is very high, given $headquarter(e, e')$ indicating "BMW, the company e, has its headquarter in Germany, the country e'". Further, such dependencies may exist only for some particular entities – e.g. not every entity that has its headquarter in Germany contains the word `munich` but some other words representing other cities in Germany. Instead of modeling dependencies between all observable variables (words and structure elements) directly, TRM models variables as being dependent on hidden topic-related variables.

Hidden Topic-Related Variables. The hidden topic variables are captured by the template attributes *topic indicator* b, *topic proportion* θ and *topic-word assignment* z. For these templates, we need object skeletons that also range over the topics T. In particular, b is instantiated with entities and topics in the skeletons $\mathcal{O}(b) = \{(e, t)|e \in \mathcal{V}_E, t \in T\}$. It is a binary-valued function such that for an entity e, $b_t(e) = 1$ indicates that t is a topic of e. The vector of all topic indicator variables of e is $\mathbf{b}(e) = \langle b_{t_1}(e), \ldots, b_{t_K}(e)\rangle$.

While $\mathbf{b}(e)$ is useful to determine which topics are present for an entity e, it does not capture sufficient information to model the probabilities of entity words. Following the tradition of topic modeling, θ is introduced for modeling entity words through a distribution of topics. While θ is defined over the same skeletons, it is different to b in that it is real-valued: for an entity e, $\theta_t(e)$ returns a

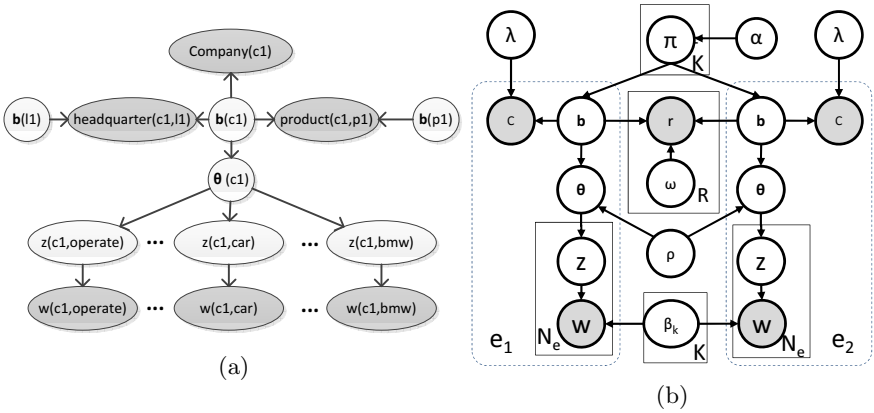


Fig. 3. (a) Template-based representation around the entity $c1$ with observed variables (dark) and hidden topic-related variables (light). (b) The generative process for two entities shown in plate notation.

real number and $\boldsymbol{\theta}(e) = \langle \theta_{t_1}(e), \ldots, \theta_{t_K}(e) \rangle$ defines a per-entity *topic distribution* such that $\sum_{t \in T} \theta_t(e) = 1$.

The semantics of the per-entity *topic-word assignment* is the same as in LDA. To capture this, we define a template attribute z that has the same skeleton as w. However, instead of a binary value, it returns a topic for an entity-word pair, i.e., $z(e, v) = t$ indicates that the word v associated with e belongs to topic t.

Fig. 3-a depicts variables obtained by instantiating the templates with information about the entity `c1`.

2.3 Probabilistic Dependencies in TRM

Central to TRM is the assumption that *given the topic indicator vector* of an entity, all its random variables derived from class and relation attributes are *conditionally independent*. That is, instead of capturing dependencies between these structure variables directly, we propose to use hidden topics. We want to capture that when entities exhibit structural resemblances, their topics shall also be similar. Vice versa, given some topics, some structure elements are more likely to be observed than others. We introduce the model parameters $\boldsymbol{\lambda}$ and $\boldsymbol{\omega}$ to capture the quantity of how much a particular structure variable depends on a topic (pair of topics).

First, we consider that the probability of observing an entity e belonging to a class c depends on its topic indicator vector $\mathbf{b}(e)$. We model this as a template factor captured by a logistic sigmoid function defined over a linear combination of topic indicators, i.e.,

$$p(c(e) \mid \mathbf{b}(e)) = \sigma(\boldsymbol{\lambda}_c^T \mathbf{b}(e)) = \sigma\left(\sum_{t_i \in T} b_{t_i}(e)\ \lambda_{ci}\right) \tag{1}$$

where $\sigma(x) = \frac{1}{1+e^{-x}}$ is the logistic sigmoid mapping values from $[-\infty, +\infty]$ to $[0, 1]$ and $\boldsymbol{\lambda}$ is a global parameter represented as a $|\mathcal{V}_C| \times K$ matrix. Each element λ_{ci} in the vector $\boldsymbol{\lambda}_c$ represents the strength of dependency between the class c and topic t_i.

Similarly, the probability of observing a relation $r(e_1, e_2)$ is modeled via logistic regression over the topic indicator vectors $\mathbf{b}(e_1)$ and $\mathbf{b}(e_2)$. A template factor over $r(e_1, e_2)$, $\mathbf{b}(e_1)$ and $\mathbf{b}(e_2)$ is defined as

$$p(r(e_1, e_2) \mid \mathbf{b}(e_1), \mathbf{b}(e_2)) = \sigma(\mathbf{b}(e_1)^T \boldsymbol{\omega}_r \mathbf{b}(e_2)) \tag{2}$$

where $\mathbf{b}(e_1)^T \boldsymbol{\omega}_r \mathbf{b}(e_2) = \sum_{t_k, t_l \in T} b_{t_k}(e_1) b_{t_l}(e_2) \omega_{rkl}$ and $\boldsymbol{\omega}_r$ is a $K \times K$ matrix. For any given two entities e_1 and e_2, where e_1 has the topic indicator t_k and e_2 has t_l, the weight of observing a relation r between these two entities is given as the value of the cell (k, l) of the matrix $\boldsymbol{\omega}_r$ denoted as ω_{rkl}.

Further, we employ the topic parameters θ and z to bring words into this picture. We want to capture that given some topics, some words are more likely than others. This part essentially follows the idea of topic modeling behind LDA. The only difference is that while LDA defines the topic proportion $\boldsymbol{\theta}(e)$ over all topics, $\boldsymbol{\theta}(e)$ here is defined only over the topics captured by the corresponding topic indicator vector $\mathbf{b}(e)$, i.e., topics not in $\mathbf{b}(e)$ have no density in $\boldsymbol{\theta}(e)$ (this creates a sparsity of topics similar to focused topic modeling [14]). To capture this, we introduce the template factor $p(\boldsymbol{\theta}(e) \mid \mathbf{b}(e))$.

This is an important design decision: On one hand, in order to handle the sparsity of dependencies that occur between the structure variables and the topics, the topics indicated in $\mathbf{b}(e)$ determines the probability of observing structure for the entity. On the other hand, the topic proportions of the entity in $\boldsymbol{\theta}(e)$ is governed by the topic indicator vector $\mathbf{b}(e)$ which in turn is determined by the structure around the entity.

2.4 Generative Process

We specify the full joint distribution for TRM and the generative process so that we can infer hidden variables from observed data in $\mathcal{G}$. First, we start with the vector $\mathbf{b}$ that specifies binary topic indicators for each entity. We use a prior distribution over the possible values in $\mathbf{b}$ in order to capture our initial uncertainty about the parameters. Obtaining such a prior is possible with the *Indian Buffet Process (IBP)*, which is a non-parametric Bayesian process used to generate latent features via a Beta-Bernoulli distribution [17]. IBP assumes that each entity e possesses a topic t with probability π_t, and that topic indicators in $\mathbf{b}(e)$ are then generated independently. Under this model, the probabilities of the topics are given as $\pi = \{\pi_1, ..., \pi_K\}$, and each π_t follows a Beta distribution with hyperparameter α, i.e., $p(\pi_t \mid \alpha) = Beta(\alpha/K, 1)$. Then, for an entity e, each topic indicator value is sampled from a Bernoulli distribution as $p(b_t(e) \mid \pi_t) = Bernoulli(\pi_t)$. IBP can be utilized for both finite and infinite number of topics [18]. Using infinite number of topics dynamically has great benefits in the case of an unsupervised topic learning so that number of topics gets larger whenever the need arises. However, in our case we are mostly interested in the

words of the entity to be distributed only to those topics selected for the classes and relations that entity has (no matter how many words the entity has, because we want to bias the topics according to structure). That's why, for the purpose of this work, we set the number of topics to a fixed K so that variational inference can be applied to learn the model in the exponential family [18]. For $\boldsymbol{\theta}(e)$, we set a Dirichlet prior over the topics just like in LDA. However, instead of using a uniform hyperparameter, we parametrize the Dirichlet with topic indicators $\mathbf{b}(e)$: $p(\boldsymbol{\theta}(e) \mid \mathbf{b}(e), \rho) = Dirichlet(\rho \mathbf{b}(e))$. As the number of selected topics varies according to $\mathbf{b}(e)$, each entity will have a different density of topic proportions. For the topic index z and word attribute w, the same process as defined for LDA involving the hyperparameter β is used. We arrive at the following generative process (see Fig. 3-b):

1. For each topic $t = 1, 2, .., K$:
   (a) Draw $\pi_t \mid \alpha \sim Beta(\alpha/K, 1)$.
2. For each entity e:
   (a) For each topic t:
      i. Draw $b_t(e) \mid \pi_t \sim Bernoulli(\pi_t)$
   (b) For each class c of entity e:
      i. Draw $c(e)$ using Eq. 1
   (c) Draw $\boldsymbol{\theta}(e) \mid \rho \sim Dir(\rho \mathbf{b}(e))$
   (d) For each word v of entity e:
      i. Draw topic index $z(e, v) \mid \boldsymbol{\theta}(e) \sim Mult(\boldsymbol{\theta}(e))$
      ii. Draw word $w(e, v) \mid z(e, v), \beta_{1:K} \sim Mult(\beta_{z(e,v)})$
3. For each pair of entities e, e':
   (a) For each relation $r \in \{r \mid r(e, e'), r(e', e) \in \mathcal{R}\}$:
      i. Draw $r(e, e')$ or $r(e', e)$ using Eq. 2

2.5 Learning

We propose to learn the posterior distribution of the hidden topic-related variables $\mathbf{b}, \boldsymbol{\theta}, \boldsymbol{\pi}$ and $\mathbf{z}$ conditioned on the observed variables $\mathbf{w}, \mathbf{c}$ and $\mathbf{r}$ via *variational Bayesian learning* [19]. Intuitively, the variational method approximates the posterior distribution of p by another simpler distribution q. In particular, through *mean field approximation* [19], we have q as a distribution that is fully-factorized over the hidden variables indexed by the free variational parameters $\boldsymbol{\nu}, \boldsymbol{\gamma}, \boldsymbol{\phi}$ and $\boldsymbol{\tau}$ for Bernoulli, Dirichlet, Multinomial and Beta distribution, respectively:

$$q(\mathbf{b}, \boldsymbol{\pi}, \boldsymbol{\theta}, \mathbf{z} \mid \boldsymbol{\nu}, \boldsymbol{\gamma}, \boldsymbol{\phi}, \boldsymbol{\tau}) = \prod_{t=1}^{K} \left[q(\pi_t \mid \tau_{t1}, \tau_{t2}) \prod_{e \in \mathcal{V}_E} q(b_t(e) \mid \nu_t(e)) \right]$$
$$\prod_{e \in \mathcal{V}_E} \left[q(\boldsymbol{\theta}(e) \mid \boldsymbol{\gamma}(e)) \prod_{(e,v) \in \mathcal{O}(w)} q(z(e, v) \mid \boldsymbol{\phi}(e, v)) \right] \tag{3}$$

These variational parameters are then fit such that q is close to the true posterior of p, where closeness is measured by the KL-divergence, $KL(q \parallel p)$. Because the decomposition $\log p(\mathbf{c}, \mathbf{r}, \mathbf{w}) = \mathrm{KL}(q \parallel p) + \mathcal{L}(q)$ and $\mathrm{KL}(q \parallel p) \geq 0$

hold [19], minimizing the KL-divergence is equivalent to maximizing the term $\mathcal{L}(q)$, the variational lower bound on the log marginal likelihood. The learning problem can then be expressed as optimizing

$$\{\tau, \nu, \gamma, \phi\} = \arg\max_{\{\tau,\nu,\gamma,\phi\}} \mathcal{L}(q) \tag{4}$$

For this, we use the 2-steps *variational Bayesian EM algorithm*. It takes the fixed hyperparameters α and ρ and an initial choice of the model parameters β, λ and ω as inputs. Then, it iteratively updates the variational parameters τ, ν, γ and ϕ until convergence in the E-step. Then, for fixed values of the variational parameters, the model parameters β, λ and ω are iteratively computed in the M-step. Thus, parameters are updated until convergence within the two steps, and both steps are run until convergence in the outer loop of the EM.

Variational E-Step. Update equations for this step can be obtained by setting the derivative of $\mathcal{L}(q)$ equal to zero. For each topic $t \in K$, we compute τ_{t1} and τ_{t2} of the Beta distribution as

$$\tau_{t1} = \frac{\alpha}{K} + \sum_{(e,t)\in\mathcal{O}(\boldsymbol{b})} \nu_t(e) \tag{5}$$

$$\tau_{t2} = 1 + |\mathcal{O}(\boldsymbol{b})| - \sum_{(e,t)\in\mathcal{O}(\boldsymbol{b})} \nu_t(e) \tag{6}$$

The update of $\nu_t(e)$ is given as $\nu_t(e) = \frac{1}{1+e^{\vartheta_t(e)}}$ where

$$\vartheta_t(e) = \vartheta_\tau + \sum_{c\in C(e)} \vartheta_c + \sum_{r(e,e')\in\mathcal{R}} \vartheta_{r1} + \sum_{r(e',e)\in\mathcal{R}} \vartheta_{r2} + \vartheta_\gamma \tag{7}$$

The update in Eq. 7 has five different parts. The contribution from the Beta prior can be computed by $\vartheta_\tau = \Psi(\tau_{t1}) - \Psi(\tau_{t2})$ where $\Psi(\cdot)$ is the digamma function. For each class $c \in C(e)$ the contribution to the update is given by

$$\vartheta_c = (1 - \sigma(\boldsymbol{\lambda}_c^T \boldsymbol{\nu}(e)))\lambda_{ct} \tag{8}$$

If the entity is the source of a relation, i.e., we have $r(e, e')$, the contribution is

$$\vartheta_{r1} = (1 - \sigma(\boldsymbol{\nu}(e)^T \boldsymbol{\omega}_r \boldsymbol{\nu}(e')))\omega_{rt.}\boldsymbol{\nu}(e') \tag{9}$$

or if it is the target, i.e. $r(e', e)$, the contribution is

$$\vartheta_{r2} = (1 - \sigma(\boldsymbol{\nu}(e')^T \boldsymbol{\omega}_r \boldsymbol{\nu}(e)))\omega_{r.t}\boldsymbol{\nu}(e') \tag{10}$$

and ϑ_γ is updated by

$$\vartheta_\gamma = \rho\ (\Psi(\gamma_t(e)) - \Psi(\sum_{t'} \gamma_{t'}(e))) \tag{11}$$

The variational Dirichlet parameter $\gamma_t(e)$ is

$$\gamma_t(e) = \rho\nu_t(e) + \sum_{(e,v)\in\mathcal{O}(w)} \phi_t(e, v) \tag{12}$$

The contribution to the update in Eq. 12 includes the variational multinomial $\phi_t(e, v)$ and also the variational parameter ν_t of the corresponding topic indicator b_t, i.e., $q(b_t(e) \mid \nu_t(e)$. This is the direct result of parameterizing the Dirichlet distribution with the topic indicator vector of each entity instead of using a non-informative prior α as in LDA.

The updates for the variational multinomial $\phi_t(e, v)$ is identical to that in variational inference for LDA [15]:

$$\phi_t(e, v) \propto \exp\{\log\beta_{tv} + \Psi(\gamma_t(e)) - \Psi(\sum_{t'} \gamma_{t'}(e))\} \tag{13}$$

where $\phi_t(e, v) = q(z(e, v) = t)$.

Variational M-Step. The update for the topic parameter β is the same as in LDA because also here, the words are conditionally dependent on β and z.

In order to fit the parameters λ and ω of the logistic regression defined by Eq. 1 and 2, respectively, we employ gradient-based optimization. At each iteration, we perform updates using the gradient

$$\nabla_{\lambda_{ct}} = \sum_{e \in \mathcal{V}_E} (1 - \sigma(\boldsymbol{\lambda}_c^T \boldsymbol{\nu}(e)))\nu_t(e) \tag{14}$$

for each class c and topic t and

$$\nabla_{\omega_{rtt'}} = \sum_{r(e,e') \in \in \mathcal{R}} (1 - \sigma(\boldsymbol{\nu}(e)^T \boldsymbol{\omega}_r \boldsymbol{\nu}(e')))\omega_{rtt'}\nu_{t'}(e') \tag{15}$$

for each relation r and topics t and t'.

These gradients cannot be used directly since they are only calculated for positive observations of classes and relations. For the unobserved cases ($r(e, e) = 0$) a regularization penalty is applied so the updates decreases ω and λ in each iteration for the topics controlled by the Beta-Bernoulli prior of b. This also introduces sparsity of the weights in ω and λ according to the topics selected in b. In particular, let ε be the number of observations (e.g. entities) for which the class membership is unknown such that $c(e) = 0$ and $\overline{\boldsymbol{\pi}}$ be a topic-indicator vector set to be the mean of the Beta-distributed variable π, $\overline{\boldsymbol{\pi}} = \frac{\boldsymbol{\tau}_1}{\boldsymbol{\tau}_1 + \boldsymbol{\tau}_2}$. Then, the regularization for $\boldsymbol{\lambda}_c$ is $\mathfrak{R}_{\boldsymbol{\lambda}_c} = -\varepsilon(\sigma(\boldsymbol{\lambda}_c^T \overline{\boldsymbol{\pi}}))\overline{\boldsymbol{\pi}}$. Similarly, let ζ be the number of observations where a particular relation is unknown (i.e. $r(e, e') = 0$). Then, the regularization term for $\boldsymbol{\omega}_r$ is $\mathfrak{R}_{\boldsymbol{\omega}_r} = -\zeta(\sigma(\overline{\boldsymbol{\pi}}^T \boldsymbol{\omega}_r \overline{\boldsymbol{\pi}})\overline{\boldsymbol{\pi}}$.

3 Experiments

First, we aim to obtain an initial understanding of the (1) *quality of the topic model* produced by TRM. Then, we provide a quantitative analysis of TRM by comparing its performance to state-of-the-art TM and SRL approaches w.r.t. the (2) *object clustering* and (3) *link predication* tasks, respectively.

Datasets. We use a subset of *DBpedia* containing 20,094 entities described by 112 distinct classes and 49 different types of relations. All attribute values are

treated as textual information and put into bags of words. The resulting vocabulary comprises 26,109 unique words after stop word removal. We also employ the *DBLP*[1] dataset. The abstract and title of the papers are treated as textual data. In addition, authors and conferences and their relations to papers are taken into account. We use a subset of papers that belong to the fields of database, data mining, information retrieval and artificial intelligence. In total, there are 28,569 paper, 28,702 author and 20 conference entities, and a vocabulary comprising 11,771 unique words.

3.1 Topic Analysis

A useful application of TRM is to understand the data. Fig. 2 displays the top words of four selected topics using the learned $\boldsymbol{\beta}$ parameter. Words are ranked by $score(v,t) = \beta_{tv}\left(\log\beta_{tv} - \frac{1}{K}\sum_{t'}\log\beta_{t'v}\right)$, which intuitively, assigns high scores to those words that are characteristic for a topic, relative to all other topics. We can clearly observe that structure elements (classes and relations) have an influence on the topics and the words that are ranked high for these topics. For example, t_1 has top words from entities of the type `organization` whereas t_2 captures words related to `person`. In particular, t_1 and t_2 have top words from those organization and person entities that are involved in the `keyPerson` relation. In fact, TRM not only exploits structure information for topic modeling but also explicitly models the strength of dependencies between topics and structure elements through the ω and λ parameters.

3.2 Link Prediction

Note that TRM captures the joint distribution over variables representing topics and structure elements. Thus, not only are topics dependent on structure elements but also vice versa, the existence of certain topics (and their words) can be used to infer that some structure elements are more likely than others. Here, we evaluate the effectiveness of using TRM topics for link prediction – based on the design and implementation used for the previous C^3 experiment [12]. We created training data using the `author` relations between papers and authors in *DBLP* and the `starring` relations between movies and actors in *DBpedia*. Then, this data is divided into a training and test set, with test data set to be 2, 4 and $\frac{3}{4}$ times the amount of training data.

However, it should be noted that the task of link prediction in SRL is different from LP for documents (e.g. [4, 5, 8]) in which topic similarity of documents is only distinctive feature. In the former each relation (e.g.starring,author) is characterized differently by its weights to features in a linear model like SVM. Thats why supervised TMs are not directly applicable on this task. Thus, in this experiment we show how the topic features based on ω of a relation are distinctive for link prediction beyond some base features such as the ones employed in C^3. Also note that, unlike other supervised TMs, TRM distinguishes different types of relations and a separate ω matrix is used for every relation, which assigns

[1] `http://www.informatik.uni-trier.de/ley/db/`

cross-topic weights. Link direction is also considered as the matrix omega is asymmetric.

Methods. We compare TRM against MLN [11] and C^3 [12]. To train the MLN, we use the open source Alchemy[2] implementation and adopt the rules as described in the Alchemy's tutorial for link predication. In order to predict links between two entities, C^3 employs SVM along with a set of features including Jaccard similarity computed from textual values of the two entities, words shared by the entities and adjacent nodes connected to the entities. LibSVM[3] is used to train a nu-SVM with a RBF kernel. The value of nu is set experimentally between 0.1 and 0.5. To use TRM for link prediction, we consider the combination of C^3 and the topics inferred by TRM. Namely, TRM provides two additional features that are then used by C^3. The first feature is a topic-based similarity score defined as $sim_r(e_1, e_2) = \sum_{t,t'} \sum_{v \in V_M} p(v \mid \beta_t)\, \omega_{rtt'}\, p(v \mid \beta_{t'})$ where V_M is the set of words e_1 and e_2 have in common. Instead of using these shared words only, we also use all the words in e_1 and e_2 to calculate a second feature using the formula above.

Table 1. Precision, recall and accuracy results for link prediction on DBLP and DBpedia

	DBLP(1-4)			DBLP(1-2)			DBLP(3-4)		
	Prec.	Rec.	Acc.	Prec.	Rec.	Acc.	Prec.	Rec.	Acc.
MLN	50.46	74.75	50.05	49.53	71.33	49.23	51.9	76.14	52.31
C^3	55.51	76.92	57.69	56.09	71.87	55.14	58.13	78.12	59.80
TRM	**66.03**	**84.84**	**67.34**	**65.98**	**83.87**	**69.53**	**68.83**	**85.54**	**71.25**
	DBpedia(1-4)			DBpedia(1-2)			DBpedia(3-4)		
	Prec.	Rec.	Acc.	Prec.	Rec.	Acc.	Prec.	Rec.	Acc.
MLN	50.72	**81.7**	51.08	50.4	**83.05**	50.5	51.45	**79.61**	52.03
C3	57.14	66.67	54.95	56.52	66.10	54.04	55.88	65.51	54.95
TRM	**72.71**	78.43	**69.04**	**70.58**	74.38	**67.84**	**71.68**	74.20	**67.28**

Results. We present the overall performance in Table 1. Also considering true negatives as depicted in Fig.4, we observe that while MLN can provide high recall for positive labeled data (achieves best recall for DBpedia in one setting), it does not perform well for negative labeled data. C^3 performs better than MLN in terms of precision and accuracy and also, achieves higher true negative rate. However, C^3's performance could clearly be improved when using TRM features in additional: TRM outperforms both baselines in terms of precision and accuracy and achieves average recall comparable to MLN. Also, it is more superior than these baselines in handling negative labeled data. The second feature provided by TRM captures the topical correlation between all words of the entities. It was particularly helpful in eliminating false negatives, i.e., two entities not correlated topic-wise w.r.t. ω_r are mostly not linked. The topic-based similarity calculated from matching words further helps to find true positives.

[2] `http://alchemy.cs.washington.edu/`
[3] `http://www.csie.ntu.edu.tw/~cjlin/libsvm/`

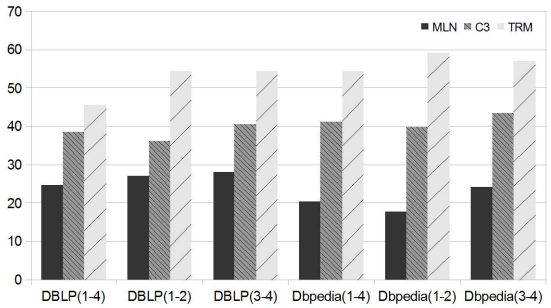


Fig. 4. True negative rate for DBLP and DBpedia

Table 2. Precision and normalized mutual information (NMI) results for object clustering on DBLP and DBpedia

Method/	Paper(%)		Author(%)		Venue(%)	
Metric	*Acc.*	*NMI*	*Acc.*	*NMI*	*Acc.*	*NMI*
LDA	47.22	15.97	–	–	–	–
TMBP-RW	69.11	45.24	74.67	65.61	65.79	67.48
TMBP-Reg	78.21	58.42	88.55	71.17	75.02	64.01
TRM	**89.35**	**65.51**	**93.44**	**78.16**	**87.73**	**75.11**

(a) DBLP

Method/	Movie(%)	
Metric	*Acc.*	*NMI*
LDA	58.71	22.29
TMBP-RW	57.10	27.65
TMBP-Reg	62.33	31.84
TRM	**71.57**	**44.28**

(b) DBpedia

3.3 Object Clustering

For DBPedia, we use entities of the type `movie`. Following the experiment performed previously [9], we use the six labels in DBLP representing various computer science fields as clusters. The clustering result is evaluated by comparing the label of each paper with the topic learned from the data.

Methods. We compare TRM to three TM approaches. As the most relevant baselines, we use two methods for learning topics from heterogeneous networks [9]: one model is learned with biased random walk (TMBP-RW) and the other results from biased regularization (TMBP-Reg). TMBP-RW propagates topic probabilities through the network via random walk, while TMBP-Reg achieves topic propagation through regularizing a statistical topic model with two generic terms. A previous experiment [9] has already shown that incorporating heterogeneous structure information as performed by these baselines helps to outperform clustering results of several existing (TM) approaches. For brevity, we thus include only the results of the standard LDA model [15]. Since LDA cannot be directly applied to heterogeneous information networks, we only use the bag-of-words representation of entities and ignore structure information.

Results. Table 2 shows the average results obtained from 10 test runs. TMBP-RW outperforms LDA on DBLP and is comparable to LDA on DBPedia. TMBP-Reg slightly outperforms TMBP-RW on both datasets. This suggests that exploiting structure information as supported by TMBP-RW and TMBP-Reg, leads to better results than LDA, which only considers word co-occurrences.

TRM leads to further improvements on both datasets by incorporating the effects of specific classes and relations on topics. For DBpedia for instance, TRM automatically infers that the structure elements `distributor` and `country` have a strong discriminative effect on assigning objects to the correct clusters.

4 Related Work

Related to our work, there are TM approaches proposed for homogeneous networks such as NetPLSA [2], Pairwise-Link-LDA [3], Nubbi [5], author-topic models [1], latent topic models for hypertext [6], citation networks [7] and relational topic models [8]. The major distinction between these models and TRM is that they consider a homogeneous network structure with only few types of entities and relations. In addition, more related to TRM, there are approaches over heterogeneous networks [9, 7] which utilize specific regularization functions to fit the topics to the underlying network structure. In general, as the network becomes more heterogeneous (i.e. more than two types of relations), more complex topic models are needed to capture complex correlations between the topic and structural variables. TRM mainly addresses this in a principled way by introducing sparsity of topics via topic indicators to create specific bias of topics towards structure information, i.e., classes and relations. In fact these approaches can be regarded as the extension of previous supervised topic models (e.g. [20–23]) to the networks in which observed variables are the relations instead of some tags or annotations. Also one similar work to ours is Type-LDA [24] which aims to discover the clusters of observed relation tuples and their associated textual expressions. However, that work only has a narrow focus on relation extraction in NLP and does not address the discovery of the correlations occurring in the whole data graph.

SRL works such as probabilistic relational models (PRM) [10] and MLN [11] learn graphical models using relational information. As discussed, this training is costly when the dependency structure is complex and the number of variables is high – which is particularly the case when a large amount of text is involved. We propose the use of hidden topic variables to reduce this complexity. To combine SRL with topic models, FoldAll [25] uses constraints given as first-order rules in a MLN to bias the topics by training a MRF. Although biasing the topics according to structure information can be accomplished through MLN, this approach does not capture correlations between topics and structure elements (e.g. predicate of rules). In addition, the number of groundings in the MLN rules poses a problem for FoldAll, since each grounding is represented as an indicator function in the corresponding topic model. TRM is unique in terms of using the topics as a low-dimensional abstraction to capture the correlations between the topics and classes/relations (i.e $\boldsymbol{\lambda}$ and $\boldsymbol{\omega}$).

5 Conclusion

We presented TRM, a novel combination of TM and SRL to learn topics from text-rich structured data. It captures dependencies between words in textual

and structured data through hidden topic variables in a *template-based model* constructed according to the underlying data structure. It represents a novel approach for automatically using heterogeneous structure information for learning topics as well as using topics to perform SRL tasks. In experiments, we show that compared to existing TM approaches, TRM is more effective in exploiting structure information. It reveals and exploits varying level of dependencies between topics and specific classes and relations, resulting in higher performance for both object clustering and link prediction.

As future work we plan to explore the extension of TRM to even richer generative models, such as time-varying and hierarchical topic models. In addition, potential application areas of TRM in the field of text-rich databases are manyfold. In particular, for selectivity estimation of structural queries comprising string predicates, TRM provides *a synopsis of the database* by capturing the topics and their correlations with classes and relations. This way, any structural query can be interpreted as a probability distribution, from which the query result size is estimated. We also consider TRM as being useful for *keyword search on structured data*. In existing work, keywords are mapped to database elements and connections between these keyword elements are discovered based on the relations given in the schema to compute structured results. The ranking of these results is separated from that computation. Instead of using the schema for discovering connections and a separate model for ranking, TRM can serve as a "probabilistic schema", capturing connections that are most probable. Hence, it can be used as a holistic model both for result computation and ranking based on their probability.

Acknowledgments. This research is partially supported by Siemens / DAAD Postgraduate Program under grant number A/10/94300.

References

1. Rosen-Zvi, M., Griffiths, T.L., Steyvers, M., Smyth, P.: The author-topic model for authors and documents. In: UAI, pp. 487–494 (2004)
2. Mei, Q., Cai, D., Zhang, D., Zhai, C.: Topic modeling with network regularization. In: WWW, pp. 101–110 (2008)
3. Nallapati, R., Ahmed, A., Xing, E.P., Cohen, W.W.: Joint latent topic models for text and citations. In: KDD, pp. 542–550 (2008)
4. Liu, Y., Niculescu-Mizil, A., Gryc, W.: Topic-link lda: joint models of topic and author community. In: ICML, p. 84 (2009)
5. Chang, J., Boyd-Graber, J.L., Blei, D.M.: Connections between the lines: augmenting social networks with text. In: KDD, pp. 169–178 (2009)
6. Gruber, A., Weiss, Y., Rosen-Zvi, M.: Hidden topic markov models. Journal of Machine Learning Research - Proceedings Track 2, 163–170 (2007)
7. Zeng, J., Cheung, W.K., Hung Li, C., Liu, J.: Multirelational topic models. In: ICDM, pp. 1070–1075 (2009)
8. Chang, J., Blei, D.M.: Relational topic models for document networks. Journal of Machine Learning Research - Proceedings Track 5, 81–88 (2009)

9. Deng, H., Han, J., Zhao, B., Yu, Y., Lin, C.X.: Probabilistic topic models with biased propagation on heterogeneous information networks. In: KDD, pp. 1271–1279 (2011)
10. Getoor, L., Taskar, B.: Introduction to statistical relational learning. MIT Press (2007)
11. Richardson, M., Domingos, P.: Markov logic networks. Machine Learning 62(1-2), 107–136 (2006)
12. Namata, G., Kok, S., Getoor, L.: Collective graph identification. In: KDD, pp. 87–95 (2011)
13. Singla, P., Domingos, P.: Entity resolution with markov logic. In: ICDM, pp. 572–582 (2006)
14. Williamson, S., Wang, C., Heller, K.A., Blei, D.M.: The ibp compound dirichlet process and its application to focused topic modeling. In: ICML, pp. 1151–1158 (2010)
15. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent dirichlet allocation. Journal of Machine Learning Research 3, 993–1022 (2003)
16. Koller, D., Friedman, N.: Probabilistic graphical models. MIT Press (2009)
17. Ghahramani, Z., Griffiths, T., Sollich, P.: Bayesian nonparametric latent feature models. Bayesian Statistics 8, 1–25 (2007)
18. Doshi-Velez, F., Miller, K., Gael, J.V., Teh, Y.W.: Variational inference for the indian buffet process. Journal of Machine Learning Research - Proceedings Track, 137–144 (2009)
19. MacKay, D.: Information theory, inference, and learning algorithms. Cambridge Univ. Pr. (2003)
20. Blei, D.M., McAuliffe, J.D.: Supervised topic models. In: NIPS (2007)
21. Rubin, T.N., Chambers, A., Smyth, P., Steyvers, M.: Statistical topic models for multi-label document classification. Machine Learning 88(1-2), 157–208 (2012)
22. Blei, D.M., Jordan, M.I.: Modeling annotated data. In: SIGIR, pp. 127–134 (2003)
23. Yakhnenko, O., Honavar, V.: Multi-modal hierarchical dirichlet process model for predicting image annotation and image-object label correspondence. In: SDM, pp. 281–294 (2009)
24. Yao, L., Haghighi, A., Riedel, S., McCallum, A.: Structured relation discovery using generative models. In: EMNLP, pp. 1456–1466 (2011)
25. Andrzejewski, D., Zhu, X., Craven, M., Recht, B.: A framework for incorporating general domain knowledge into latent dirichlet allocation using first-order logic. In: IJCAI, pp. 1171–1177 (2011)

A Confidentiality Model for Ontologies

Piero A. Bonatti and Luigi Sauro

Dept. of Electrical Engineering and Information Technologies
Università di Napoli "Federico II"

Abstract. We illustrate several novel attacks to the confidentiality of knowledge bases (KB). Then we introduce a new confidentiality model, sensitive enough to detect those attacks, and a method for constructing secure KB views. We identify safe approximations of the background knowledge exploited in the attacks; they can be used to reduce the complexity of constructing secure KB views.

1 Introduction

There is ample evidence of the need for knowledge confidentiality measures. OWL and the LOD paradigm are increasingly being used to encode the private knowledge of companies and public organizations. Linked open government data include potentially sensitive information, e.g. related to health. Medical records are annotated with semantic metadata based on SNOMED. FOAF assertions and other semantic description of social networks may affect the privacy of individuals. In all of these cases, semantic web techniques help in linking different knowledge sources and extract implicit information, thereby increasing security and privacy risks. Even the authors of *public* ontologies may want to hide some axioms to capitalize on their formalization efforts. See [8] for further motivations. In order to tackle the confidentiality requirements arising from these scenarios, several approaches have been proposed. The most popular security criterion is that the published view of the knowledge base should not entail any secret sentence (we call it *simple confidentiality model*). However, there exist attacks that cannot be blocked this way. The user may exploit various sources of background knowledge and metaknowledge to reconstruct the hidden part of the knowledge base. This paper contributes to the area of knowledge base confidentiality in several ways:

(i) It highlights some vulnerabilities of the approaches that can be found in the literature, including attacks based on meta-reasoning (Sec. 3).

(ii) It introduces a stronger confidentiality model that takes both object-level and meta-level background knowledge into account (Sec. 4), and it defines a method for computing secure knowledge views (Sec. 5) that generalizes some previous approaches.

(iii) It proposes a safe approximation of background metaknowledge (Sec. 6 and 7).

(iv) It investigates the computational complexity of constructing secure knowledge base views with our methodology (Sec. 7).

The paper is closed by a discussion of related work (Sec. 9), and conclusions. Some proofs are omitted due to space limitations.

H. Alani et al. (Eds.): ISWC 2013, Part I, LNCS 8218, pp. 17–32, 2013.

2 Preliminaries on Description Logics

We assume the reader to be familiar with description logics, and refer to [1] for all definitions and results. We assume a fixed, denumerable signature Σ specifying the names of *concepts*, *roles*, and *individuals*. Our framework is compatible with any description logic DL that enjoys compactness (needed by Theorem 6) and has decidable reasoning problems (e.g., $\mathcal{ALC}$, $\mathcal{EL}$, $\mathcal{SHIQ}$, etc.). We simply assume that our reference logical language $\mathcal{L}$ is generated from Σ by the grammar of the selected logic DL. By *axioms*, we mean members of $\mathcal{L}$, unless stated otherwise. A *knowledge base* is any subset of $\mathcal{L}$.[1]

Recall that axioms are expressions of the form $C \sqsubseteq D$, $R \sqsubseteq S$, $C(a)$, and $R(a, b)$ where C, D are concept expressions, R, S are role expressions, and a, b are individual constants. In some DL, an individual constant a may occur also in a *nominal*, that is, a concept expression $\{a\}$ denoting the singleton containing a. The axioms involving $\sqsubseteq$ are called *inclusions* (or *subsumptions*), while $C(a)$ and $R(a, b)$ are called *assertions*. In the simplest case, C and R are first order predicates and assertions are actually standard first-order atomic formulae. Inclusions are syntactic variants of logical implications.

The notion of *logical consequence* is the classical one; for all $K \subseteq \mathcal{L}$, the logical consequences of K will be denoted by $Cn(K)$ ($K \subseteq Cn(K) \subseteq \mathcal{L}$).

3 A Simple Confidentiality Model

The most natural way of preserving confidentiality in a knowledge base KB is checking that its answers to user queries do not entail any secret. Conceptually, the queries of a user u are answered using u's view KB_u of the knowledge base, where KB_u is a maximal subset of KB that entails no secret. In order to illustrate some possible attacks to this mechanism, let us formalize the above *simple confidentiality model* (SCM).[2] It consists of: the knowledge base KB ($KB \subseteq \mathcal{L}$); a set of users U; a *view* $KB_u \subseteq KB$ for each $u \in U$; a set of *secrecies* $S_u \subseteq \mathcal{L}$ for each $u \in U$. Secrecies are axioms that may or may not be entailed by KB; if they do, then they are called *secrets* and must not be disclosed to u. Revealing that a secrecy is *not* entailed by KB is harmless [4]. For example, there is no need to protect the information that someone is *not* having chemotherapy.

A view KB_u is *secure* iff $Cn(KB_u) \cap S_u = \emptyset$. A view KB_u is *maximal secure* if it is secure and there exists no K such that $KB_u \subset K \subseteq KB$ and $Cn(K) \cap S_u = \emptyset$.

Attacks Using Object-Level Background Knowledge. Frequently, part of the domain knowledge is not axiomatized in KB, therefore checking that $Cn(KB_u) \cap S_u = \emptyset$ does not suffice in practice to protect confidentiality. For example, suppose that there is one secret $S_u = \{OncologyPatient(John)\}$ and $KB_u = \{SSN(John, 12345), SSN(user123, 12345), OncologyPatient(user123)\}$. KB_u does not entail $OncologyPatient(John)$, so according to the SCM model KB_u is secure. However, it is common knowledge that a SSN uniquely identifies a person, then the user can infer that $John = user123$, and hence the secret.

In other examples, the additional knowledge used to infer secrets may be stored in a public ontology or RDF repository, and confidentiality violations may be automated.

[1] Real knowledge bases are finite, but this restriction is not technically needed until Sec. 7.

[2] This usage of term "*model*" is common in Security & Privacy.

Attacks to Complete Knowledge. Suppose the attacker knows that *KB* has complete knowledge about a certain set of axioms. Then the attacker may be able to reconstruct some secrets from the "I don't know" answers of a maximal secure view KB_u.

Example 1. Consider an organization's knowledge base that defines a concept *Employee* and a role *works_for* that describes which employees belong to which of the n departments of the company, $d_1, \ldots, d_n$. The *KB* consists of assertions like:

$$Employee(e) \qquad (1) \qquad\qquad works\_for(e, d_i) \qquad (2)$$

where we assume that each employee e belongs to exactly one department d_i. A user u is authorized to see all assertions but the instances of (2) with $i = n$, because d_n is a special department, devoted to classified projects. So S_u (the set of secrecies for u) is the set of all assertions $works\_for(e, d_n)$.

Note that there is one maximal secure view KB_u. It consists of all instances of (1), plus all instances of (2) such that $i \neq n$. Clearly, KB_u is secure according to SCM (because $Cn(KB_u) \cap S_u = \emptyset$). However, observe that $works\_for(e, d_n) \in Cn(KB)$ iff $Employee(e) \in Cn(KB_u)$ and for all $i = 1, \ldots, n$, $works\_for(e, d_i) \notin Cn(KB_u)$ (that is, the members of d_n are all the employees that apparently work for no department). Using this property (based on the knowledge that for each employee e, *KB* contains exactly one assertion $works\_for(e, d_i)$) and the knowledge of the protection mechanism (i.e. maximal secure views), that we assume to be known by attackers by *Kerchoff's principle*, a smart user can easily identify all the members of d_n. □

In practice, it is not hard to identify complete knowledge. A hospital's *KB* is expected to have complete knowledge about which patients are in which ward; a company's *KB* is likely to encode complete information about its employees, etc.

Some approaches filter query answers rather than publishing a subset of *KB* [7, 14, 16]. We call our abstraction of this method *simple answer confidentiality model* (SACM). It is obtained from the SCM by replacing the views $KB_u \subseteq KB$ with *answer views* $KB^a_u \subseteq Cn(KB)$. The difference is that KB^a_u is not required to be a subset of *KB* and—conceptually—KB^a_u may be infinite. KB^a_u is *secure* iff $Cn(KB^a_u) \cap S_u = \emptyset$.

The reader may easily verify that the SACM is vulnerable to the two kinds of attacks illustrated for the SCM. It is also vulnerable to a third kind of attacks, illustrated below.

Attacks to the Signature. Suppose the user knows the signature of *KB* well enough to identify a symbol σ that does not occur in *KB*. First assume that σ is a concept name. It can be proved that:

Proposition 1. *If KB^a_u is a maximal secure answer view and σ is a concept name not occurring in KB, then for all secrecies $C \sqsubseteq D \in S_u$, $KB^a_u \models C \sqcap \sigma \sqsubseteq D$ iff $KB \models C \sqsubseteq D$.*

The problem is that although $C \sqcap \sigma \sqsubseteq D$ does not entail the secret inclusion $C \sqsubseteq D$, still a smart user knows that the former inclusion cannot be proved unless *KB* entails also the latter (then maximal secure answer views generally fail to protect secrets). This attack can be easily adapted to the case where σ is a role name. In practice, it is not necessary to be sure that σ does not occur in *KB*. The attacker may make a sequence of educated guesses (say, by trying meaningless long strings, or any word that is clearly

unrelated to the domain of the *KB*); after a sufficient number of trials, the majority of answers should agree with the "real" answer with high probability. Rejecting queries whose signature is not contained in *KB*'s signature mitigates this kind of attacks but it leaks *KB*'s signature and it does not provide a complete solution: Any σ occurring in *KB* that is logically unrelated to *C* and *D* can be used for a similar attack.

4 A Meta-safe Confidentiality Model

In this section we introduce a confidentiality model that makes the vulnerabilities illustrated above visible, by taking into account object- and meta-level background knowledge. A *bk-model* $\mathcal{M} = \langle KB, U, f, \langle S_u, PKB_u, BK_u \rangle_{u \in U} \rangle$ consists of a knowledge base $KB \subseteq \mathcal{L}$, a set of users U, plus:

- a *filtering function* $f : \wp(\mathcal{L}) \times U \to \wp(\mathcal{L})$, mapping each knowledge base K and each user u on a view $f(K, u) \subseteq Cn(K)$;
- for all $u \in U$:
  - a finite set of secrecies $S_u \subseteq \mathcal{L}$;
  - a set of axioms $BK_u \subseteq \mathcal{L}$, encoding the users' object-level knowledge;
  - a set of *possible knowledge bases* $PKB_u \subseteq \wp(\mathcal{L})$ (users' metaknowledge).[3]

The view of *KB* released to a user u is $f(KB, u)$. We adopt *PKB* because at this stage we do not want to tie our framework to any specific metalanguage. *PKB* represents the knowledge bases that are compatible with the user's metaknowledge.

Definition 1. *A filtering function f is* secure *(w.r.t. $\mathcal{M}$) iff for all $u \in U$ and all $s \in S_u$, there exists $K \in PKB_u$ such that:*

1. $f(K, u) = f(KB, u)$;
2. $s \notin Cn(K \cup BK_u)$.

Intuitively, if f is safe according to Def. 1, then no user u can conclude that any secret s is entailed by the *KB* she is interacting with—enhanced with the object-level background knowledge BK_u—for the following reasons: By point 1, *KB* and K have the same observable behavior, and K is a possible knowledge base for u since $K \in PKB_u$; therefore, as far as u knows, the knowledge base might be K. Moreover, by point 2, K and the object-level background knowledge BK_u do not suffice to entail the secret s.

In the rest of the paper we tacitly assume that no secret is violated a priori, that is, for all secrets $s \in S_u$ there exists $K \in PKB_u$ such that $s \notin Cn(K \cup BK_u)$.[4] *Moreover, in order to improve readability, we shall omit the user u from subscripts and argument lists* whenever u is irrelevant to the context.

The attacks discussed in Section 3 can be easily formalized in this setting; so, in general, the maximal secure views of SCM are not secure according to Def. 1.

[3] In practice, bk-models are finite, and filterings computable, but no such assumption will be technically needed until Sec. 7.

[4] Conversely, no filtering function can conceal a secret that is already known by the user.

Example 2. Example 1 can be formalized in our model as follows: The set of secrecies S is the set of all assertions $works\_for(e, d_n)$; $BK = \emptyset$ and PKB is the set of all the knowledge bases K that consist of assertions like (1) and (2), and such that for each axiom $Employee(e)$, K contains exactly one corresponding axiom $works\_for(e, d_i)$ and viceversa. The filtering function f maps each $K \in PKB$ on the maximal subset of K that entails none of S's members, that is, $f(K) = K \setminus S$ (by definition of PKB).

Note that f is injective over PKB, so condition 1 of Def. 1 is satisfied only if $K = KB$. So, if KB contains at least one secret, then the conditions of Def. 1 cannot be satisfied, that is, maximal secure SCM views are not secure in our model. Indeed, KB can be reconstructed from the secure view by observing that $KB = f(KB) \cup \{works\_for(e, d_n) \mid Employee(e) \in f(KB) \wedge \forall i = 1, \ldots, n, works\_for(e, d_i) \notin f(KB)\}$. □

Similarly, the formalizations of the other attacks yield injective filtering functions (the details are left to the reader).

5 A Meta-secure Query Answering Mechanism

In this section we introduce a *secure filtering function*. It is formulated as an iterative process based on a *censor*, that is a boolean function that decides for each axiom whether it should be obfuscated to protect confidentiality. The iterative construction manipulates pairs $\langle X^+, X^- \rangle \in \wp(\mathcal{L}) \times \wp(\mathcal{L})$ that represent a meta constraint on possible knowledge bases: we say that a knowledge base K *satisfies* $\langle X^+, X^- \rangle$ iff K entails all the sentences in X^+ and none of those in X^- (formally, $Cn(K) \supseteq X^+$ and $Cn(K) \cap X^- = \emptyset$).

Let PAX (the set of *possible axioms*) be the set of axioms that may occur in the knowledge base according to the user's knowledge, i.e. $PAX = \bigcup_{K' \in PKB} K'$. Let $\nu = |PAX| + 1$ if PAX is finite and $\nu = \omega$ otherwise; let $\alpha_1, \alpha_2, \ldots, \alpha_i, \ldots$ be any enumeration of PAX ($i < \nu$).[5] The secure view construction for a knowledge base K in a bk-model $\mathcal{M}$ consists of the following, inductively defined sequence of pairs $\langle K_i^+, K_i^- \rangle_{i \geq 0}$:

- $\langle K_0^+, K_0^- \rangle = \langle \emptyset, \emptyset \rangle$, and for all $1 \leq i < \nu$, $\langle K_{i+1}^+, K_{i+1}^- \rangle$ is defined as follows:
  - if $censor_{\mathcal{M}}(K_i^+, K_i^-, \alpha_{i+1}) = true$ then let $\langle K_{i+1}^+, K_{i+1}^- \rangle = \langle K_i^+, K_i^- \rangle$;
  - if $censor_{\mathcal{M}}(K_i^+, K_i^-, \alpha_{i+1}) = false$ and $K \models \alpha_{i+1}$ then $\langle K_{i+1}^+, K_{i+1}^- \rangle = \langle K_i^+ \cup \{\alpha_{i+1}\}, K_i^- \rangle$;
  - otherwise let $\langle K_{i+1}^+, K_{i+1}^- \rangle = \langle K_i^+, K_i^- \cup \{\alpha_{i+1}\} \rangle$.

Finally, let $K^+ = \bigcup_{i<\nu} K_i^+$, $K^- = \bigcup_{i<\nu} K_i^-$, and $f_{\mathcal{M}}(K, u) = K^+$.

Note that the inductive construction aims at finding maximal sets K^+ and K^- that (i) partly describe what does / does not follow from K (as K satisfies $\langle K^+, K^- \rangle$ by construction), and (ii) do not trigger the censor (the sentences α_{i+1} that trigger the censor are included neither in K^+ nor in K^-, cf. the induction step).

In order to define the censor we need an auxiliary definition that captures all the sentences that can be entailed with the background knowledge BK and the meta-knowledge PKB enriched by a given constraint $\langle X^+, X^- \rangle$ analogous to those adopted in the iterative construction: Let $Cn_{\mathcal{M}}(X^+, X^-)$ be the set of all axioms $\alpha \in \mathcal{L}$ such that

$$\text{for all } K' \in PKB \text{ such that } K' \text{ satisfies } \langle X^+, X^- \rangle, \alpha \in Cn(K' \cup BK)\,. \tag{3}$$

[5] We will show later how to restrict the construction to finite sequences, by approximating PAX.

Now the censor is defined as follows: For all $X^+, X^- \subseteq \mathcal{L}$ and $\alpha \in \mathcal{L}$,

$$censor_{\mathcal{M}}(X^+, X^-, \alpha) = \begin{cases} true & \text{if there exists } s \in S \text{ s.t. either } s \in Cn_{\mathcal{M}}(X^+ \cup \{\alpha\}, X^-) \\ & \text{or } s \in Cn_{\mathcal{M}}(X^+, X^- \cup \{\alpha\}); \\ false & \text{otherwise.} \end{cases} \quad (4)$$

In other words, the censor checks whether telling either that α is derivable or that α is not derivable to a user aware that the knowledge base satisfies $\langle X^+, X^- \rangle$, restricts the set of possible knowledge bases enough to conclude that a secret s is entailed by the knowledge base and the background knowledge encoded by BK and PKB.

Note that the censor obfuscates α_{i+1} if *any* of its possible answers entail a secret, independently of the actual contents of K (the two possible answers "yes" and "no" correspond to conditions $s \in Cn_{\mathcal{M}}(X^+ \cup \{\alpha\}, X^-)$ and $s \in Cn_{\mathcal{M}}(X^+, X^- \cup \{\alpha\})$, respectively). In this way, roughly speaking, the knowledge bases that entail s are given the same observable behavior as those that don't. Under a suitable continuity assumption on $Cn_{\mathcal{M}}$, this enforces confidentiality:

Theorem 1. *If* $Cn_{\mathcal{M}}(KB^+, KB^-) \subseteq \bigcup_{i<\nu} Cn_{\mathcal{M}}(KB_i^+, KB_i^-)$*, then* $f_{\mathcal{M}}$ *is secure w.r.t.* $\mathcal{M}$*.*

Proof. Let u and s be arbitrary members of U and S_u, respectively. We have to show that there exists a $K \in PKB_u$ satisfying the two conditions of Def. 1. Let $\langle KB_i^+, KB_i^- \rangle_{i<\nu}$ be the sequence underlying the construction of $f_{\mathcal{M}}(KB, u)$. By construction, for all $i < \nu$, $s \notin Cn_{\mathcal{M}}(KB_i^+, KB_i^-)$. Moreover, by the continuity hypothesis, $Cn_{\mathcal{M}}(KB^+, KB^-) \subseteq \bigcup_{i<\nu} Cn_{\mathcal{M}}(KB_i^+, KB_i^-)$, where $KB^+ = \bigcup_{i<\nu} KB_i^+$ and $KB^- = \bigcup_{i<\nu} KB_i^-$. It follows that $s \notin Cn_{\mathcal{M}}(KB^+, KB^-)$. Then, by definition of $Cn_{\mathcal{M}}$, there exists $K \in PKB_u$ such that:

$$Cn(K) \supseteq KB^+ \quad (5) \qquad Cn(K) \cap KB^- = \emptyset \quad (6) \qquad s \notin Cn(K \cup BK_u)\,. \quad (7)$$

Since (7) is the second condition of Def. 1, we are only left to show the first one, that is, $f_{\mathcal{M}}(K, u) = f_{\mathcal{M}}(KB, u)$. It suffices to prove by induction on i that for all $i < \nu$,

$$\langle K_i^+, K_i^- \rangle = \langle KB_i^+, KB_i^- \rangle\,. \quad (8)$$

The base case is trivial. Induction step ($i > 0$): By induction hypothesis, (8) holds for $i-1$, therefore $censor_{\mathcal{M}}(K_{i-1}^+, K_{i-1}^-, \alpha_i) = censor_{\mathcal{M}}(KB_{i-1}^+, KB_{i-1}^-, \alpha_i)$. If the censors are true, then (8) follows directly from the induction hypothesis. If the censor is false, then α_i belongs to $KB^+ \cup KB^-$; note that K and KB agree on these formulae, by (5) and (6), so both knowledge bases insert α_i into the same element of the i-th pair and (8) holds. □

Examples of the behavior of $f_{\mathcal{M}}$ are deferred until Sec.7.

6 Approximating Background Knowledge

Of course, the actual confidentiality of a filtering $f(KB, u)$ depends on a careful definition of the user's background knowledge, that is, PKB_u and BK_u. If background knowledge is not exactly known, as it typically happens, then it can be safely approximated by *overestimating* it. More background knowledge means larger BK_u and smaller PKB_u, which leads to the following comparison relation $\leq_k$ over bk-models:

Definition 2. *Given two bk-models* $\mathcal{M} = \langle KB, U, f, \langle S_u, PKB_u, BK_u\rangle_{u\in U}\rangle$ *and* $\mathcal{M}' = \langle KB', U', f', \langle S'_u, PKB'_u, BK'_u\rangle_{u\in U'}\rangle$*, we write* $\mathcal{M} \leq_k \mathcal{M}'$ *iff*

1. $KB = KB'$, $U = U'$, $f = f'$, *and* $S_u = S'_u$ *(for all* $u \in U$*);*
2. *for all* $u \in U$, $PKB_u \supseteq PKB'_u$ *and* $BK_u \subseteq BK'_u$.

The next proposition proves that a bk-model $\mathcal{M}$ can be safely approximated by any $\mathcal{M}'$ such that $\mathcal{M} \leq_k \mathcal{M}'$:

Proposition 2. *If* f *is secure w.r.t.* $\mathcal{M}'$ *and* $\mathcal{M} \leq_k \mathcal{M}'$*, then* f *is secure w.r.t.* $\mathcal{M}$.

Consequently, a generic advice for estimating *BK* consists in including as many pieces of relevant knowledge as possible, for example:

(i) modelling as completely as possible the integrity constraints satisfied by the data, as well as role domain and range restrictions and disjointness constraints;

(ii) including in *BK* all the relevant public sources of formalized relevant knowledge (such as ontologies and triple stores).

While object-level background knowledge is dealt with in the literature, the general metaknowledge encoded by *PKB* is novel. Therefore, the next section is focussed on some concrete approximations of *PKB* and their properties.

7 Approximating and Reasoning about Possible Knowledge Bases

In this section, we investigate the real world situations where *the knowledge base KB is finite* and *so are all the components of bk-models* (U, S_u, BK_u, PKB_u); then we focus on PKB_u that contain only finite knowledge bases. Consequently, $f_{\mathcal{M}}$ will turn out to be decidable and we will study its complexity under different assumptions.

A language for defining *PKB* is a necessary prerequisite for the practical implementation of our framework and a detailed complexity analysis of $f_{\mathcal{M}}$. Here we express *PKB* as the set of all theories that are contained in a given set of *possible axioms PAX* [6] and satisfy a given, finite set *MR* of *metarules* like:

$$\alpha_1, \ldots, \alpha_n \Rightarrow \beta_1 \mid \ldots \mid \beta_m \quad (n \geq 0, m \geq 0)\,, \tag{9}$$

where all α_i and β_j are in $\mathcal{L}$ ($1 \leq i \leq n$, $1 \leq j \leq m$). Informally, (9) means that if *KB* entails $\alpha_1, \ldots, \alpha_n$ then *KB* entails also some of $\beta_1, \ldots, \beta_m$. Sets of similar metarules can be succintly specified using *metavariables*; they can be placed wherever individual constants may occur, that is, as arguments of assertions, and in nominals. A metarule with such variables abbreviates the set of its *ground instantiations*: Given a $K \subseteq \mathcal{L}$, let $ground_K(MR)$ be the ground (variable-free) instantiation of *MR* where metavariables are uniformly replaced by the individual constants occurring in K in all possible ways.

Example 3. Let $MR = \{\, \exists R.\{X\} \Rightarrow A(X) \,\}$, where X is a metavariable, and let $K = \{\, R(a, b) \,\}$. Then $ground_K(MR) = \{\, (\exists R.\{a\} \Rightarrow A(a)),\ (\exists R.\{b\} \Rightarrow A(b)) \,\}$. □

If r denotes rule (9), then let $body(r) = \{\alpha_1, \ldots, \alpha_n\}$ and $head(r) = \{\beta_1, \ldots, \beta_m\}$. We say r is *Horn* if $|head(r)| \leq 1$. A set of axioms $K \subseteq \mathcal{L}$ *satisfies* a ground metarule r if either $body(r) \nsubseteq Cn(K)$ or $head(r) \cap Cn(K) \neq \emptyset$. In this case we write $K \models_m r$.

[6] Differently from Sec. 5, here *PKB* is defined in terms of *PAX*.

Example 4. Let A, B, C be concept names and R be a role name. The axiom set $K = \{A \sqsubseteq \exists R.B, A \sqsubseteq C\}$ satisfies $A \sqsubseteq \exists R \Rightarrow A \sqsubseteq B \mid A \sqsubseteq C$ but not $A \sqsubseteq \exists R \Rightarrow A \sqsubseteq B$. □

Moreover, if K satisfies all the metarules in $ground_K(MR)$ then we write $K \models_m MR$. Therefore the formal definition of *PKB* now becomes:

$$PKB = \{K \mid K \subseteq PAX \wedge K \models_m MR\}\,. \tag{10}$$

In accordance with Prop. 2, we approximate *PAX* in a conservative way. We will analyze two possible definitions:

1. $PAX_0 = KB$ (i.e., as a minimalistic choice we only assume that the axioms of *KB* are possible axioms; of course, by Prop. 2, this choice is safe also w.r.t. any larger *PAX* where *at least* the axioms of *KB* are regarded as possible axioms);
2. $PAX_1 = KB \cup \bigcup_{r \in ground_{KB}(MR)} head(r)$.

Remark 1. The latter definition is most natural if metarules are automatically extracted from *KB* with rule mining techniques, that typically construct rules using material from the given *KB* (then rule heads occur in *KB*).

Example 5. Consider again Example 1. The user's metaknowledge about *KB*'s completeness can be encoded with:

$$Employee(X) \Rightarrow works\_for(X, d_1) \mid \ldots \mid works\_for(X, d_n)\,, \tag{11}$$

where X is a metavariable. First let $PAX = PAX_1$. The secure view $f_{\mathcal{M}}(KB)$ depends on the enumeration order of *PAX*. If the role assertions $works\_for(e, d_i)$ precede the concept assertions $Employee(e)$, then, in a first stage, the sets KB_j^+ are progressively filled with the role assertions with $d_i \neq d_n$ that belong to *KB*, while the sets KB_j^- accumulate all the role assertions that do not belong to *KB*. In a second stage, the sets KB_j^+ are further extended with the concept assertions $Employee(e)$ such that e does not work for d_n. The role assertions $works\_for(e, d_n)$ of *KB* and the corresponding concept assertions $Employee(e)$ are neither in KB^+ nor in KB^-. Note that the final effect is equivalent to removing from *KB* all the axioms referring to the individuals that work for d_n. Analogously, in [7] the individuals belonging to a specified set are removed from all answers.

Next suppose that the role assertions $works\_for(e, d_i)$ follow the concept assertions $Employee(e)$, and that each $works\_for(e, d_i)$ follows all $works\_for(e, d_k)$ such that $k < i$. Now all the assertions $Employee(e)$ of *KB* enter KB^+, and all axioms $works\_for(e, d_i)$ with $i < n - 1$ enter either KB^+ or KB^-, depending on whether they are members of *KB* or not. Finally, the assertions $works\_for(e, d_i) \in Cn(KB)$ with $i \in \{n-1, n\}$ are inserted neither in KB^+ nor in KB^-, because the corresponding instance of (11) with $X = e$ has the body in KB^+ and the first $n - 2$ alternatives in the head in KB^-, therefore a negative answer to $works\_for(e, d_{n-1})$ would entail the secret $works\_for(e, d_n)$ by (11). This triggers the censor for all assertions $works\_for(e, d_{n-1})$. Summarizing, with this enumeration ordering it is possible to return the complete list of employees; the members of d_n are protected by hiding also which employees belong to d_{n-1}.

Finally, let $PAX = PAX_0$. In this case, all possible knowledge bases are subsets of *KB*; the latter contains exactly one assertion $works\_for(e, d_{i(e)})$ for each employee e.

Then, in order to satisfy (11), every $K \in PKB$ containing $Employee(e)$ must contain also $works\_for(e, d_{i(e)})$. It follows that $f_{\mathcal{M}}$ must remove all references to the individuals e that work for d_n, as it happens with the first enumeration of PAX_1. □

Definition 3. *A bk-model $\mathcal{M}$ is* canonical *if for all users $u \in U$, PAX_u is either PAX_0 or PAX_1 and PKB_u is defined by (10) for a given MR_u. Moreover, $\mathcal{M}$ is* in a description logic DL *if for all $u \in U$, all the axioms in KB, PKB_u, BK_u, and S_u belong to DL.*

The size of PAX_0 and PAX_1[7] is polynomial in the size of $KB \cup MR$, therefore PKB is finite and exponential in the size of $KB \cup MR$. Finiteness implies the continuity hypothesis on $Cn_{\mathcal{M}}$ of Theorem 1, and hence (using Theorem 1 and Prop. 2):

Theorem 2. *If $\mathcal{M}$ is canonical, then $f_{\mathcal{M}}$ is secure with respect to all $\mathcal{M}' \leq_k \mathcal{M}$.*

Proof. Since $\mathcal{M}$ is canonical, for all $u \in U$, PKB_u is finite and $\nu = |PAX_u| + 1 < \omega$. By construction, the sets KB_i^+ and KB_i^- in the sequence $\langle KB_i^+, KB_i^- \rangle_{i<\nu}$ grow monotonically with i, so $\bigcup_{i<\nu} KB_i^+ = KB_{\nu-1}^+$ and $\bigcup_{i<\nu} KB_i^- = KB_{\nu-1}^-$. Moreover, $Cn_{\mathcal{M}}$ is monotonic in both arguments, so $\bigcup_{i<\nu} Cn_{\mathcal{M}}(KB_i^+, KB_i^-) = Cn_{\mathcal{M}}(KB_{\nu-1}^+, KB_{\nu-1}^-)$. It follows that

$$Cn_{\mathcal{M}}(\bigcup_{i<\nu} KB_i^+, \bigcup_{i<\nu} KB_i^-) = Cn_{\mathcal{M}}(KB_{\nu-1}^+, KB_{\nu-1}^-) = \bigcup_{i<\nu} Cn_{\mathcal{M}}(KB_i^+, KB_i^-)\,,$$

that is, the continuity hypothesis of Theorem 1 is satisfied. Then, by Theorem 1, $f_{\mathcal{M}}$ is secure with respect to $\mathcal{M}$, and by Prop. 2, $f_{\mathcal{M}}$ is secure with respect to all $\mathcal{M}' \leq_k \mathcal{M}$. □

First we analyze the complexity of constructing the secure view $f_{\mathcal{M}}(KB)$ when the underlying description logic is tractable, like $\mathcal{EL}$ and DL-lite for example.

Lemma 1. *If the axioms occurring in MR and K are in a DL with tractable subsumption and instance checking, then checking $K \models_m MR$ is:*

1. *in P if either MR is ground or there exists a fixed bound on the number of distinct variables in MR;*
2. *coNP-complete otherwise.*

Proof. Point 1: $K \models_m MR$ can be checked as follows: For each $r \in ground_K(MR)$ and all axioms $\alpha \in body(r) \cup head(r)$ check whether $K \models_m r$ by verifying whether there exists either $\alpha \in body(r)$ such that $\alpha \notin Cn(K)$, or $\alpha \in head(r)$ such that $\alpha \in Cn(K)$. The cost of each test $K \models_m r$ is polynomial in the size of MR and K since membership in $Cn(K)$ is in P by hypothesis. The number of iterations is polynomial in the size of MR and K, too, because the hypothesis that the number of variables in r is bounded implies that $|ground_K(MR)|$ is polynomial in the size of MR and K.

Point 2: (Membership) The complementary test $K \not\models_m MR$ can be carried out in two steps: first guess an $r \in MR$ and a substitution σ that maps each metavariable in r on an individual constant occurring in K; second, check whether $K \models_m r\sigma$ does *not* hold. Checking $K \models_m r\sigma$ is in P (cf. point 1), so $K \not\models_m MR$ can be checked in nondetermistic polynomial time, and hence the original problem ($K \models_m MR$) is in coNP.

[7] We assume here and in the following complexity results that axiom sets—and hence KBs—have a natural encoding as strings that determine their size.

Hardness follows by reducing to $K \not\models_m MR$ the clause subsumption problem: *Given two clauses (i.e. two sets of literals) G and H, is there a substitution σ such that $G\sigma \subseteq H$?* (if the answer is "yes" then *G subsumes H*). The problem is still NP-complete if all literals are positive, terms are function-free, and predicate arity is bounded by 2. Let $G = \{p_1, \ldots, p_n\}$ and H be two clauses satisfying these assumptions. Let $K = H$ (i.e. K is a set of assertions whose concept names and role names are the unary and binary predicates of H, respectively, and whose individual constants are the terms occurring in H). Let $MR = \{p_1, \ldots, p_n \Rightarrow\}$, where the terms occurring in G and not in H are regarded as metavariables. Now G subsumes H iff there exists a substitution σ such that $G\sigma \subseteq H$, iff there is an instance $r \in ground_K(MR)$ such that $K \not\models_m r$, iff $K \not\models_m MR$. □

With Lemma 1, one can prove the following two lemmas.

Lemma 2. *Let $\mathcal{M}$ range over canonical bk-models. If $\mathcal{M}$, s, X^+, and X^- are in a DL with tractable subsumption/instance checking, and the number of distinct variables in MR is bounded by a constant, then checking whether $s \in Cn_{\mathcal{M}}(X^+, X^-)$ is:*

1. *in P if MR is Horn and $PAX = PAX_1$;*
2. *coNP-complete if either MR is not Horn or $PAX = PAX_0$.*

Proof. Point 1: By standard logic programming techniques, a minimal $K \subseteq PAX$ satisfying MR and entailing X^+ can be obtained with the following PTIME construction:

$$K_0 = X^+, \quad K_{i+1} = K_i \cup \bigcup \{ head(r) \mid r \in ground_{K_i}(MR) \wedge body(r) \subseteq Cn(K_i) \}. \qquad (12)$$

This sequence reaches its limit after at most $|PAX|$ iterations. Then $s \in Cn_{\mathcal{M}}(X^+, X^-)$ holds iff either $s \in K_{|PAX|}$ or $K_{|PAX|} \cap X^- \neq \emptyset$. Both tests are in P since $K_{|PAX|} \subseteq PAX$.

Point 2: Membership in coNP is straightforward ($s \notin Cn_{\mathcal{M}}(X^+, X^-)$ can be checked by guessing a $K \subseteq PAX$ that satisfies $\langle X^+, X^- \rangle$ and such that $s \notin Cn(K \cup BK)$). To prove hardness first assume that $PAX = PAX_0$. For each given 3-SAT instance, encode its n propositional variables and their negation with $2n$ concept names P_i and $\bar{P}_i$, respectively. Introduce a concept name C_k for each clause $c_k = l_{k,1} \vee l_{k,2} \vee l_{k,3}$. Let KB consist of all the inclusions $A \sqsubseteq P_i$ and $A \sqsubseteq \bar{P}_i$ ($1 \leq i \leq n$), plus all $L_{k,j} \sqsubseteq C_k$ s.t. $L_{k,j}$ is the encoding of l_{kj} ($j = 1, 2, 3$). Let $s = (A \sqsubseteq B)$, $BK = \emptyset$ and let MR consists of all the rules $(A \sqsubseteq P_i, A \sqsubseteq \bar{P}_i \Rightarrow)$, $(\Rightarrow L_{k,j} \sqsubseteq C_k)$, $(\Rightarrow A \sqsubseteq C_k)$. MR is Horn, and the given clause set is satisfiable iff there exists $K \subseteq KB = PAX_0$ such that $K \models_m MR$. For all such K, $s \notin Cn(K)$ because B does not occur in KB. Then the given clauses are satisfiable iff $s \notin Cn_{\mathcal{M}}(\emptyset, \emptyset)$. This proves that checking whether $s \in Cn_{\mathcal{M}}(X^+, X^-)$ is coNP-hard.

We are left to show a similar result under the assumption that MR is not Horn and $PAX = PAX_1$. Let KB, s, and BK be defined as before. Let MR be the set of all rules $(A \sqsubseteq P_i, A \sqsubseteq \bar{P}_i \Rightarrow)$, $(\Rightarrow A \sqsubseteq P_i \mid A \sqsubseteq \bar{P}_i)$, $(A \sqsubseteq \bar{L}_{k,1}, A \sqsubseteq \bar{L}_{k,2}, A \sqsubseteq \bar{L}_{k,3} \Rightarrow s)$, where each $\bar{L}_{kj}$ is the encoding of the complement of l_{kj}. Clearly, the given set of clauses is satisfied iff there exists $K \subseteq PAX_1$ such that $K \models_m MR$ and $s \notin Cn(K)$; this is equivalent to $s \notin Cn_{\mathcal{M}}(\emptyset, \emptyset)$. The theorem follows immediately. □

Lemma 3. *Let $\mathcal{M}$ be a canonical bk-model. If $\mathcal{M}$, s, X^+, and X^- are in a DL with tractable entailment problems, and there is no bound on the number of variables in the metarules of MR, then checking $s \in Cn_{\mathcal{M}}(X^+, X^-)$ is:*

1. *in P^{NP} if MR is Horn and PAX = PAX_1;*
2. *in Π_2^p if either MR is not Horn or PAX = PAX_0.*

Proof. To prove Point 1, we use the same algorithm used for Lemma 2.(1), based on the bottom-up construction defined by (12). However, due to the lack of bounds on metavariables, $ground_{K_i}(MR)$ can be exponentially large. Then the complexity of each iteration in (12) is determined with a different, nondeterministic algorithm: For each possible ground instance of a rule head (quadratically many due to arity bounds) use the NP oracle to guess an instance of the rule body and check (in polynomial time) whether it is entailed by K_i. The deterministic algorithm then runs in polynomial time using an NP oracle.

Point 2 can be proved with the naive nondeterministic algorithm that guesses a $K \subseteq PAX$ and checks whether (i) $K \in PKB$, (ii) $X^+ \subseteq Cn(K)$ and $X^- \cap Cn(K) = \emptyset$, and (iii) $s \in Cn(K \cup BK_u)$. Condition (i) can be verified by checking whether $K \models_m MR$; this test is NP-complete by Lemma 1.(2). Conditions (ii) and (iii) are in P by hypothesis. So the whole nondeterministic algorithm runs in polynomial time using an NP oracle. □

The value of $censor(X^+, X^-, \alpha)$ can be computed straightforwardly by iterating the tests $s \in Cn_{\mathcal{M}}(X^+ \cup \{\alpha\}, X^-)$ and $s \in Cn_{\mathcal{M}}(X^+, X^- \cup \{\alpha\})$ for all secrets $s \in S$. Since the set of secrets is part of the parameter $\mathcal{M}$ of the filtering function, the number of iterations is polynomial in the input and the complexity of the censor is dominated by the complexity of $Cn_{\mathcal{M}}()$. The latter is determined by Lemma 2 and Lemma 3, so we immediately get:

Corollary 1. *Let $\mathcal{M}$ be a canonical bk-model and suppose that $\mathcal{M}$, X^+, X^-, and α are in a DL with tractable entailment problems. If the number of distinct variables in MR is bounded by a constant, then computing $censor(X^+, X^-, \alpha)$ is:*

- *in P if MR is Horn and PAX = PAX_1;*
- *coNP-complete if either MR is not Horn or PAX = PAX_0.*

If there is no bound on the number of variables in the metarules of MR, then computing $censor(X^+, X^-, \alpha)$ is:

- *in P^{NP} if MR is Horn and PAX = PAX_1;*
- *in Π_2^p if either MR is not Horn or PAX = PAX_0.*

We are now ready to analyze the complexity of filtering functions:

Theorem 3. *If $\mathcal{M}$ is a canonical bk-model in a DL with tractable entailment problems, then computing $f_{\mathcal{M}}(KB)$ is:*

1. *in P if the number of distinct variables in the rules of MR is bounded, MR is Horn, and PAX = PAX_1;*
2. *P^{NP}-complete if the number of distinct variables in MR is bounded, and either MR is not Horn or PAX = PAX_0;*
3. *in P^{NP} if the variables in MR are unbounded, MR is Horn, and PAX = PAX_1;*
4. *in Δ_3^p if MR is not restricted and PAX $\in \{PAX_0, PAX_1\}$.*

Proof. Point 1 follows easily from Corollary 1: use the straightforward algorithm that iterates over all α_i in the enumeration of *PAX*, and for each of them computes the censor and checks whether $KB \models \alpha$ (if needed); since the number of iterations is polynomial in the input, the overall complexity is dominated by the complexity of evaluating the censor and KB entailments (both are tractable).

Point 2: Assume that $PAX = PAX_0$ and *MR* is Horn, the other case where $PAX = PAX_1$ and *MR* is not Horn can be proved with the techniques adopted in Lemma 2.(2). Membership in P^{NP} is straightforward, by the same argument applied in Point 1. Hardness is proved by a reduction of the maximum satisfying assignment problem which, given a set of clauses $C = \{c_1, \ldots, c_m\}$ in the variables $p_1, \ldots, p_n$, consists in finding the lexicographically maximum assignment $\mu^{msa} \in \{0,1\}^n$ that satisfies C, or 0 if C is unsatisfiable. We extend *KB* and *MR* defined at point 2 in Lemma 2 as follows: first, we add to *KB* $A \sqsubseteq C$ and the set of inclusions $A \sqsubseteq P'_i$, with $1 \leq i \leq n$. Secondly, we replace the rules $\Rightarrow A \sqsubseteq C_k$, $1 \leq k \leq m$, with $A \sqsubseteq C \Rightarrow A \sqsubseteq C_k$ and add the rules $A \sqsubseteq P'_i \Rightarrow A \sqsubseteq P_i$, with $1 \leq i \leq n$. Finally, consider an ordering of *PAX* where $\alpha_1 = A \sqsubseteq C$ and, for each $1 \leq i \leq n$, $\alpha_{i+1} = A \sqsubseteq P'_{n+1-i}$ (the rest of the ordering is not relevant).

The inclusion $A \sqsubseteq C$ plays the role of a satisfiability checker for C, that is if C is not satisfiable, then for all $K \in KB$, $A \sqsubseteq C \notin Cn(K)$. Consequently, $A \sqsubseteq C \notin f_{\mathcal{M}}(KB)$. Assume now that C is satisfiable. First of all, since for all $0 \leq i \leq n$ $KB \models \alpha_i$, then $KB^- = \emptyset$ and $\alpha_i \in KB^+$ iff $censor_{\mathcal{M}}(KB_i^+, \emptyset, \alpha_i)$ is false. Now, note that the α_i are not forced to be entailed by any rule, therefore for each $K \in PKB$ also $K \setminus \{\alpha_i\} \in PKB$. Consequently, $censor_{\mathcal{M}}(KB_i^+, \emptyset, \alpha_i)$ is false iff there exists a $K \in PKB$ such that $Cn(K) \supseteq KB_i^+ \cup \{\alpha_i\}$. In particular, this ensures that $A \sqsubseteq C \in KB^+$. Now, since *PKB* satisfies the rules $A \sqsubseteq P'_i \Rightarrow A \sqsubseteq P_i$ and encodes with the inclusions $A \sqsubseteq P_i$ all possible assignments ν that satisfy C, this means that $censor_{\mathcal{M}}(KB_i^+, \emptyset, \alpha_i)$ is false (i.e. $\alpha_i \in KB^+$) iff there exists an assignment μ such that $\mu_i = 1$ and for all $1 \leq j < i$, $\mu_j = 1$ iff $A \sqsubseteq P'_j \in KB^+$. Finally, from the fact that most significant $A \sqsubseteq P'_i$ are processed first, we have that if C is not satisfiable then $A \sqsubseteq C \notin f_{\mathcal{M}}(KB)$, otherwise $A \sqsubseteq C \in f_{\mathcal{M}}(KB)$ and $A \sqsubseteq P'_i \in f_{\mathcal{M}}(KB)$ iff $\mu_i^{msa} = 1$.

Points 3, 4 are straightforward by the same argument for membership in Point 1. □

Theorem 4. *Computing $f_{\mathcal{M}}(KB)$ over canonical $\mathcal{M}$ in a DL with ExpTime entailment (e.g. $\mathcal{ALCQO}$, $\mathcal{ALCIO}$, $\mathcal{ALCQI}$, $\mathcal{SHOQ}$, $\mathcal{SHIO}$, $\mathcal{SHIQ}$), is still in ExpTime.*

Proof. Consider any test $s \in Cn_{\mathcal{M}}(X^+, X^-)$ in the construction of $f_{\mathcal{M}}(KB, u)$ (there are two such tests for each censor evaluation). Carrying out the test for given X^+, X^-, and $s \in S_u$ can be done by brute force, iterating over all the exponentially many $K' \subseteq PAX$ (which is either PAX_0 or PAX_1 whose size is polynomial in *KB*). For each such K', we have to verify whether it belongs to *PKB*, by checking whether $K' \models_m MR$; this can be done in ExpTime by iterating over all the (ground) instances $r \in ground_K(MR)$ and checking in polynomial time whether $K' \models_m r$. Then, for all $K' \in PKB$, three ExpTime problems must be solved ($X^+ \subseteq Cn(K')$, $X^- \cap Cn(K') = \emptyset$, and $s \notin Cn(K' \cup BK_u)$). If they all succeed, $s \notin Cn_{\mathcal{M}}(X^+, X^-)$; otherwise the algorithm continues with the next $K' \subseteq PAX$. So the total cost of each censor call is exponential in the size of *KB*, *MR*, and BK_u. In order to compute $f_{\mathcal{M}}(KB)$, this cost is iterated for all combinations of secrets and axioms in *PAX*; moreover, for each iteration where the censor is false, an

additional ExpTime entailment problem is solved ($KB \models \alpha_{i+1}$). It follows that computing $f_{\mathcal{M}}(KB, u)$ is exponential in the size of KB, MR, BK_u, and S_u. □

Theorem 5. *Computing $f_{\mathcal{M}}(KB)$ over canonical $\mathcal{M}$ in $\mathcal{SROIQ}(\mathcal{D})$ is in coNP^{N2ExpTime}.*

Proof. (Hint) Use the same brute-force algorithm used in Theorem 4. □

8 Relationships with the SCM

Here we show that the meta-secure framework is a natural generalization of the SCM. The main result—roughly speaking—demonstrates that the SCM model can be essentially regarded as a special case of our framework where $PKB \supseteq \wp(KB)$ and $BK = \emptyset$. In this case $f_{\mathcal{M}}$ is secure even if $\mathcal{M}$ is not assumed to be canonical.

Theorem 6. *Let $\mathcal{M} = \langle KB, U, f_{\mathcal{M}}, \langle S_u, PKB_u, BK_u \rangle_{u \in U} \rangle$. If $PKB = \wp(KB)$, $BK = \emptyset$, and KB is finite, then*

1. *$Cn_{\mathcal{M}}(KB^+, KB^-) = \bigcup_{i<\nu} Cn_{\mathcal{M}}(KB_i^+, KB_i^-)$.*
2. *For all enumerations of PAX, the corresponding $f_{\mathcal{M}}(KB, u)$ is logically equivalent to a maximal secure view KB_u of KB according to the SCM; conversely, for all maximal secure view KB_u of KB (according to the SCM) there exists an enumeration of PAX such that the resulting $f_{\mathcal{M}}(KB, u)$ is logically equivalent to KB_u.*
3. *$f_{\mathcal{M}}$ is secure w.r.t. $\mathcal{M}$ and w.r.t. any $\mathcal{M}' = \langle KB, U, f_{\mathcal{M}}, \langle S_u, PKB'_u, BK'_u \rangle_{u \in U} \rangle$ such that $PKB' \supseteq \wp(KB)$ and $BK' = \emptyset$.*

Proof. By the first hypothesis, $PAX = KB$. As a first consequence, for all $\alpha \in PAX$, $\alpha \in Cn(KB)$, and hence, by definition of the inductive sequence $\langle KB_i^+, KB_i^- \rangle_{i<\nu}$, we have that all for all $i < \nu$, $KB_i^- = \emptyset$. As a second consequence, for all $X^+ \subseteq KB$, we have $X^+ \in PKB = \wp(KB)$. Therefore X^+ is also the least $K \in PKB$ (up to logical equivalence) such that $Cn(K) \supseteq X^+$ and $Cn(K) \cap \emptyset = \emptyset$. This fact and the second hypothesis imply (by definition of $Cn_{\mathcal{M}}$) that

$$Cn_{\mathcal{M}}(X^+, \emptyset) = Cn(X^+)\,. \tag{13}$$

As a special case, we get $Cn_{\mathcal{M}}(KB_i^+, KB_i^-) = Cn(KB_i^+)$, for all $i < \nu$. Moreover, by compactness, $Cn(\bigcup_{i<\nu} KB_i^+) = \bigcup_{i<\nu} Cn(KB_i^+)$; then Point 1 follows by:

$$Cn_{\mathcal{M}}(KB^+, KB^-) = Cn(KB^+) = Cn(\bigcup_{i<\nu} KB_i^+) = \bigcup_{i<\nu} Cn(KB_i^+) = \bigcup_{i<\nu} Cn_{\mathcal{M}}(KB_i^+, KB_i^-).$$

Now let $\alpha_1, \alpha_2, \ldots, \alpha_i, \ldots$ be any enumeration of PAX. By induction on i, it is easy to prove (using (13) and the definitions of K_{i+1}^+ and the censor) that for all $i < \nu$, $\alpha_i \notin K_i^+$ iff either $Cn(K_{i-1}^+ \cup \{\alpha_i\}) \cap S \neq \emptyset$ or $\alpha_i \in Cn(K_{i-1})$. It follows immediately that $\bigcup_{i<\nu} KB_i^+$ is logically equivalent to a maximal subset KB_u of KB that does not entail any secret. By definition, the same holds for $f_{\mathcal{M}}(KB, u) = \bigcup_{i<\nu} KB_i^+$.

Conversely, let KB_u be any maximal subset of KB that entails no secret, and let $n = |KB_u|$. Let $\alpha_1, \alpha_2, \ldots, \alpha_i, \ldots$ be any enumeration of PAX such that $KB_u = \{\alpha_1, \ldots, \alpha_n\}$ (i.e. the sentences in KB_u precede those in $KB \setminus KB_u$). As in the above paragraph, it can be verified that for all $i < \nu$, $\alpha_i \notin K^+_{i+1}$ iff either $Cn(K^+_i \cup \{\alpha_i\}) \cap S \neq \emptyset$ or $\alpha_i \in Cn(K_i)$. It follows that KB^+_n is logically equivalent to KB_u, and for all $i > n$, $KB^+_i = KB_n$. Consequently, $\bigcup_{i<\nu} KB^+_i$ is logically equivalent to KB_u, and so is $f_{\mathcal{M}}(KB, u) = \bigcup_{i<\nu} KB^+_i$. This completes the proof of Point 2.

Point 3: $f_{\mathcal{M}}$ is secure w.r.t. $\mathcal{M}$ by Theorem 1, whose hypothesis is satisfied by Point 1. It follows by Proposition 2, that $f_{\mathcal{M}}$ is also secure for all $\mathcal{M}'$ such that $\mathcal{M}' \leq_k \mathcal{M}$, which includes all $\mathcal{M}'$ that are identical to $\mathcal{M}$ with the exception of their possible knowledge bases PKB', and such that $PKB' \supseteq PKB = \wp(KB)$. □

Remark 2. Theorem 6 applies to every canonical $\mathcal{M}$ such that $MR = BK = \emptyset$, because $MR = \emptyset$ implies that $PAX_0 = PAX_1 = KB$ and hence $PKB = \wp(KB)$. This shows that the SCM can be regarded as a special case of our framework where the user has no background knowledge. Moreover, by this correpondence, one immediately obtains complexity bounds for the SCM from those for PAX_1 and Horn, bounded-variable MR.

9 Related Work

Baader et al. [2], Eldora et al. [12], and Knechtel and Stuckenschmidt [14] attach security labels to axioms and users to determine which subset of the KB can be used by each subject. These works are instances of the SCM so they are potentially vulnerable to the attacks based on background knowledge; this holds in particular for [14] that pursues the construction of maximal secure views. Similar considerations hold for [16]. Moreover, in [2, 12] axiom labels are not derived from the set of secrets; knowledge engineers are responsible for checking ex post that no confidential knowledge is entailed; in case of leakage, the view can be modified with a revision tool based on pinpointing. Our mechanism produces automatically a secure view from the secrets, instead, and decides secondary protection, i.e. which additional axioms shall be hidden for security.

Chen and Stuckenschmidt [7] adopt an instance of the SACM based on removing some individuals entirely. In general, this may be secure against metaknowledge attacks (cf. Ex. 5). However, no methodology is provided for selecting the individuals to be removed given a target set of secrets.

In [3], KB is partioned into a visible part KB_v and a hidden part KB_h. Conceptually, this is analogous to axiom labelling, cf. the above approaches. Their confidentiality methodology seems to work only under the assumption that the signatures of KB_v and KB_h are disjoint, because in strong safety they do not consider the formulae that are implied by a combination of KB_v and KB_h. Surely the axioms of KB_h whose signature is included in the signature of KB_v cannot be protected, in general. A partition-based approach is taken in [10], too. It is not discussed how to select the hidden part KB_h given a set of target secrets (which includes the issue of deciding secondary protection).

Similarly, in [15] only ex-post confidentiality verification methods are provided. In their model the equivalent of PKB is the set of all knowledge bases that include a given set of publicly known axioms $\mathcal{S} \subseteq KB$; consequently, their verification method is

vulnerable to the attacks to complete knowledge based on conditional metaknowledge (cf. Example 2 and Example 5) that cannot be encoded in their framework.

Cuenca Grau and Horrocks [9] investigate knowledge confidentiality from a probabilistic perspective: enlarging the public view should not change the probability distribution over the possible answers to a "sensitive query" Q that represents the set of secrets. In [9] users can query the knowledge base only through a pre-defined set of views (we place no such restriction, instead). A probability distribution P over the set of knowledge bases plays a role similar to metaknowledge. However, their confidentiality condition allows P to be replaced with a different P' after enlarging the public view, so at a closer look P does not really model the user's a priori knowledge about the knowledge base (that should remain constant), differently from our *PKB*.

Our method is inspired by the literature on *controlled (database) query evaluation* (CQE) based on lies and/or refusals ([4, 5, 6] etc). Technically we use *lies*, because rejected queries are not explicitly marked. However, our censor resembles the classical refusal censor, so the properties of $f_{\mathcal{M}}$ are not subsumed by any of the classical CQE methods. For example (unlike the CQE approaches that use lies), $f_{\mathcal{M}}(KB, u)$ encodes only correct knowledge, and it is secure even if users initially know a disjunction of secrets. Unlike the refusal method, $f_{\mathcal{M}}$ can handle *cover stories* because users are not told which queries are obfuscated; as an additional advantage, our method needs not to adapt existing engines to handle nonstandard answers like refusals (*mum*).

10 Discussion and Conclusions

We identified some novel vulnerabilities of those confidentiality preservation methods that do not take background knowledge into account. The new confidentiality model of Sec. 4 can detect these vulnerabilities, based on a generic formalization of object- and meta-level background knowledge. A general mechanism for constructing secure views (the filtering $f_{\mathcal{M}}$) is provably secure w.r.t. this model under a continuity assumption, and generalizes a few previous approaches (cf. Thm. 6 and Ex. 5). In order to compute secure views in practice we introduced a safe, generic method for approximating background knowledge, and a specific rule-based metalanguage. In this instantiation of the general framework $f_{\mathcal{M}}$ is always secure and its complexity can be analyzed.

If the underlying DL is tractable, then in the simplest case $f_{\mathcal{M}}$ can be computed in polynomial time. The number of variables in metarules and the adoption of a more secure approximation (PAX_0) may increase complexity up to $\mathrm{P}^{NP} = \Delta_2^p$ and perhaps Δ_3^p. The complexity of non-Horn metarules, however, can be avoided by replacing each non-Horn r with one of its Horn strengthenings: $body(r) \Rightarrow \alpha$ such that $\alpha \in head(r)$. This approximation is safe (because it restricts *PKB*), and opens the way to a systematic use of the low-complexity bk-models based on PAX_1 and Horn metarules.

For the many ExpTime-complete DL, secure view computation does not increase asymptotic complexity. So far, the best upper complexity bound for computing secure views in the description logic underlying OWL DL (i.e. $\mathcal{SROIQ}(\mathcal{D})$) is $\mathrm{coNP}^{N2ExpTime}$.

We plan to refine these complexity results and investigate different tradeoffs between information availability and computational complexity. Moreover, the idea of mining metarules from *KB* is particularly intriguing: it would be the first automated support to background knowledge approximation.

We are investigating implementations of the low-complexity frameworks (based on PAX_1 and Horn metarules) using the incremental engine versions available for Pellet and ELK to avoid repeated classifications in the iterative construction of $f_{\mathcal{M}}$. Metarule bodies can be evaluated with SPARQL. Answer set programming technologies (e.g. DLV-Hex [11]) provide interesting alternatives. Secure views are constructed off-line, so no overhead is placed on user queries, that can be answered with any standard engine. For these reasons, our approach is expected to be applicable in practice.

References

[1] Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F.: The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press (2003)
[2] Baader, F., Knechtel, M., Peñaloza, R.: A generic approach for large-scale ontological reasoning in the presence of access restrictions to the ontology's axioms. In: International Semantic Web Conference, pp. 49–64 (2009)
[3] Bao, J., Slutzki, G., Honavar, V.: Privacy-preserving reasoning on the semantic web. In: Web Intelligence, pp. 791–797. IEEE Computer Society (2007)
[4] Biskup, J., Bonatti, P.A.: Lying versus refusal for known potential secrets. Data Knowl. Eng. 38(2), 199–222 (2001)
[5] Biskup, J., Bonatti, P.A.: Controlled query evaluation for enforcing confidentiality in complete information systems. Int. J. Inf. Sec. 3(1), 14–27 (2004)
[6] Biskup, J., Bonatti, P.A.: Controlled query evaluation for known policies by combining lying and refusal. Ann. Math. Artif. Intell. 40(1-2), 37–62 (2004)
[7] Chen, W., Stuckenschmidt, H.: A model-driven approach to enable access control for ontologies. In: Wirtschaftsinformatik, Österreichische Computer Gesellschaft. books@ocg.at, vol. 246, pp. 663–672 (2009)
[8] Cuenca Grau, B.: Privacy in ontology-based information systems: A pending matter. Semantic Web 1(1-2), 137–141 (2010)
[9] Cuenca Grau, B., Horrocks, I.: Privacy-preserving query answering in logic-based information systems. In: ECAI 2008, pp. 40–44. IOS Press (2008)
[10] Cuenca Grau, B., Motik, B.: Importing ontologies with hidden content. In: Description Logics. CEUR Workshop Proceedings, vol. 477. CEUR-WS.org (2009)
[11] Eiter, T., Fink, M., Krennwallner, T., Redl, C.: Conflict-driven ASP solving with external sources. Theory and Practice of Logic Programming 12(4-5), 659–679 (2012)
[12] Eldora, M.K., Peñaloza, R.: Correcting access restrictions to a consequence more flexibly. In: Description Logics. CEUR Workshop Proc., vol. 745, CEUR-WS.org (2011)
[13] Hitzler, P., Lukasiewicz, T. (eds.): RR 2010. LNCS, vol. 6333. Springer, Heidelberg (2010)
[14] Knechtel, M., Stuckenschmidt, H.: Query-based access control for ontologies. In: Hitzler, Lukasiewicz (eds.) [13], pp. 73–87
[15] Stouppa, P., Studer, T.: Data privacy for knowledge bases. In: Artemov, S., Nerode, A. (eds.) LFCS 2009. LNCS, vol. 5407, pp. 409–421. Springer, Heidelberg (2008)
[16] Tao, J., Slutzki, G., Honavar, V.: Secrecy-preserving query answering for instance checking in $\mathcal{EL}$. In: Hitzler, Lukasiewicz (eds.) [13], pp. 195–203

Pattern Based Knowledge Base Enrichment

Lorenz Bühmann and Jens Lehmann

Universität Leipzig, Institut für Informatik, AKSW,
Postfach 100920, D-04009 Leipzig, Germany
{buehmann,lehmann}@informatik.uni-leipzig.de
http://aksw.org

Abstract. Although an increasing number of RDF knowledge bases are published, many of those consist primarily of instance data and lack sophisticated schemata. Having such schemata allows more powerful querying, consistency checking and debugging as well as improved inference. One of the reasons why schemata are still rare is the effort required to create them. In this article, we propose a semi-automatic schemata construction approach addressing this problem: First, the frequency of axiom patterns in existing knowledge bases is discovered. Afterwards, those patterns are converted to SPARQL based pattern detection algorithms, which allow to enrich knowledge base schemata. We argue that we present the first scalable knowledge base enrichment approach based on real schema usage patterns. The approach is evaluated on a large set of knowledge bases with a quantitative and qualitative result analysis.

1 Introduction

Over the past years, the quantity and size of RDF knowledge bases has significantly increased. Nevertheless, many of those knowledge bases lack sophisticated schemata and instance data adhering to those schemata. For content extracted from legacy sources, crowdsourced content, but also manually curated content, it is challenging to ensure a co-evolution of schemata and data, in particular for large knowledge bases. For this reason, there has been significant recent interest in semi-automation of schemata creation and revision based on the available instance data [7,18,31,33]. The combination of instance data and schemata allows improved querying, inference and consistency checking. In particular, in previous work [7], we investigated lightweight and efficient schema creation approaches, which can scale to large knowledge bases. Furthermore, those methods are able to work with SPARQL based access to knowledge bases, which is currently the dominating form for querying knowledge bases, which cannot easily be handled by standard OWL reasoners. The main drawback of this early work is that we were limited to learn property axioms, e.g. domain and range. In this work, we go one step further and provide an approach, which is able to handle many frequent axiom types, while still being efficient. This is achieved by following a two phase approach: First, we analyse several data repositories to detect which terminological axiom patterns are frequently used and convert those patterns

H. Alani et al. (Eds.): ISWC 2013, Part I, LNCS 8218, pp. 33–48, 2013.

to SPARQL queries, which help to find those axiom patterns in instance data. This preparation phase only needs to be performed once. Secondly, we perform a lightweight statistical analysis to actually find specific axiom candidates as suggestions for a knowledge engineer and compute a confidence score for each of them.

Example 1. As a running example for knowledge base enrichment, consider an axiom pattern[1]:

$$\mathtt{A} \equiv \mathtt{B} \sqcap \exists \mathtt{r}.\mathtt{C}$$

which can be instantiated by an axiom

$$\mathtt{SoccerPlayer} \equiv \mathtt{Person} \sqcap \exists \mathtt{team}.\mathtt{SoccerClub}$$

describing that every person which is in a team that is a soccer club, is a soccer player. Adding such an axiom to a knowledge base can have several benefits: 1.) The axioms serve as documentation for the purpose and correct usage of schema elements. 2.) They improve the application of constraint violation techniques. For instance, when using a tool such as the Pellet Constraint Validator[2] on a knowledge base with the above axiom, it would report soccer players without an associated team as violation.[3] 3.) Additional implicit information can be inferred, e.g. in the above example each person, who is in a soccer club team can be inferred to belong to the class `SoccerPlayer`, which means that an explicit assignment to that class is no longer necessary. The main purpose of our research is, therefore, to reduce the effort of creating and maintaining such schema information by providing enrichment suggestions to knowledge base maintainers.

We implemented our enrichment methods in the DL-Learner[4] framework [17] based on earlier work in [22,20]. The ORE tool [19][5] provides a graphical interface for them. Our main contributions are as follows:

- An analysis of 1392 ontologies containing approximately 20.5 million terminological axioms with respect to axiom patterns.
- Scalable retrieval and evaluation methods via SPARQL using sampling and confidence estimation.
- A manual evaluation of 11 patterns and 718 axioms in DBpedia [26].
- An open source implementation in DL-Learner.

The article is structured as follows: First, we present the overall workflow in Section 2. After that, the axiom normalisation into patterns and the estimation of their usage frequency is described in Section 3. Using [8] for converting such a

[1] We use standard description logic syntax in this paper and refer to [2] for an introduction.

[2] `http://clarkparsia.com/pellet/icv/`

[3] Under OWL semantics, this is not a violation, due to the Open World Assumption, unless we can infer from other knowledge that the player has no team.

[4] `http://dl-learner.org`

[5] `http://ore-tool.net`

pattern to a SPARQL query, we show how to suggest new candidates for axioms, which could be added to the knowledge base in Section 4. For each suggestion, we provide a confidence value based on F-Score, which is detailed in Section 5. A performance optimisation for the workflow is outlined in Section 6. In Sections 7 and 8, we present our experimental setup and evaluation results, in particular we discuss benefits as well as cases in which our approach fails to provide correct suggestions. Related work is presented in Section 9 and we conclude in Section 10.

2 Knowledge Base Enrichment Workflow

In this section, we describe the overall workflow illustrated by Figure 1.

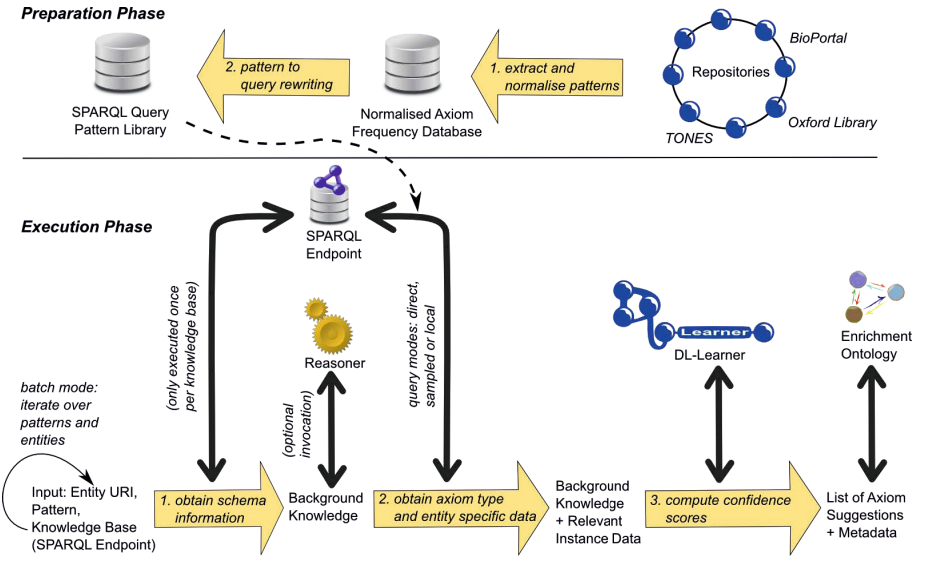


Fig. 1. Enrichment Workflow: In the preparation phase, typical axiom patterns are detected and converted to SPARQL queries, which are then used in the execution phase to learn new axiom suggestions

The *preparation phase* (upper part), described in the next section, results in an automatically compiled library of query patterns for learning frequent axioms. Herein, the frequency is determined by analysing several ontology repositories and, afterwards, applying the method in [8] for converting the patterns to SPARQL queries.

In the *execution phase*, which is an extension of previous work [7], the actual axiom suggestions are generated. To achieve this, a single algorithm run takes an axiom pattern as input and generates a set of OWL axioms as a result. It proceeds in three steps:

1. In the optional first step, SPARQL queries are used to obtain existing information about the schema of the knowledge base, in particular we retrieve

axioms which allow to construct the class hierarchy. It can be configured whether to use an OWL reasoner for inferencing over the schema or just taking explicit knowledge into account.[6] Naturally, the schema only needs to be obtained once per knowledge base and can then be re-used by all algorithms and all entities.
2. The second step consists of obtaining data via SPARQL, which is relevant for learning the considered axiom. This results in a set of axiom candidates, configured via a threshold.
3. In the third step, the score of axiom candidates is computed and the results returned.

We will explain the preparation phase and steps 2 and 3 of the execution phase in the following sections in more detail by referring to our running example.

3 Pattern Frequency Detection

For detecting patterns, a set of input OWL files is used. Each axiom in those files is then transformed to a normal form, which we call an *axiom pattern*, which is defined as structural equivalence class[7].

An axiom is transformed to a pattern as follows: Let $Sig(ax)$ be the signature of an OWL axiom ax. Let C be an ordered list of named classes, P be an ordered list of properties and I be an ordered list of individuals. This order is then extended to class expressions using an ordering over the different types of class expressions. Based on this ordering, we can re-order the elements of intersection and disjunction expressions in axioms. After that, each element of $Sig(ax)$ is replaced by a placeholder variable.

As an example, this normalisation ensures that the axiom pattern $A \sqsubseteq B \sqcap \exists r.(C)$ is equally obtained from both $Father \sqsubseteq Person \sqcap \exists hasChild.Male$ and $Carnivore \sqsubseteq \exists eat.Meat \sqcap Animal$.

Naturally, there is no unique way to define patterns. For instance, in the above approach, we focus on patterns containing a single axiom. It would also be possible to detect sets of axioms, which combined result in typical usage patterns. At this stage, we did not do this due to scalability reasons: The algorithm needs to be able to read hundreds of potentially large files and, in a later stage, the generated SPARQL queries need to run on knowledge bases with billions of facts.

4 Data Retrieval

Usually, we have to run 3 SPARQL queries to obtain all data for computing the score by means of precision and recall: one query for the number of instances

[6] Note that the OWL reasoner only loads the schema of the knowledge base and, therefore, this option worked even in cases with several hundred thousand classes in our experiments using the HermiT reasoner.

[7] `http://www.w3.org/TR/owl2-syntax/#Structural_Specification`

of the left hand side ($|A|$), one for the instance count of the right hand side ($|B \sqcap \exists\ p.C|$), and another one to get the number of instances contained in the intersection of both ($|A \sqcap B \sqcap \exists\ p.C|$). Based on that information, we can compute precision P as $P = \frac{|A \sqcap B \sqcap \exists p.C|}{|B \sqcap \exists p.C|}$ and recall R as $R = \frac{|A \sqcap B \sqcap \exists p.C|}{|A|}$ in our example.

The transformation function defined in [8] applied to $A \sqcap B \sqcap \exists\ p.C$ (the intersection) leads to the following SPARQL query pattern:

```
?x a <A> .
?x a <B> .
?x <r> ?s0 .
?s0 a <C> .
```

Once having converted an axiom pattern into a SPARQL query pattern, in a next step we need to replace entities of the query pattern with variables V, resulting in another query pattern. Usually, the left hand side of the pattern represents the named classes in our knowledge base. For this reason, we can also iterate over all classes. This is not formally necessary, but in practice it splits up a large problem into subproblems, each of which requires less expensive SPARQL queries. Assuming that D is the current class, we obtain the following query:

```
?x a <D> .
?x a ?cls1 .
?x ?p ?s0 .
?s0 a ?cls2.
```

After that, we group and project the results to all entities which were replaced by variables in the previous step (`?p`, `?cls0`, `?cls1` in this case), and count the frequency for each combination. This results in a set of pattern instantiations and frequency counts. In Example 1, the corresponding is:

```
SELECT ?p ?cls0 ?cls1 (COUNT(DISTINCT(?x) as ?cnt)) WHERE
{ ?x a <D>.
  ?x a ?cls0.
  ?x ?p ?s0 .
  ?s0 a ?cls1
} GROUP BY ?p ?cls0 ?cls1 ORDER BY DESC(?cnt)
```

We assume that we do this for all three required queries (left hand side, right hand side, intersection).

5 Pattern Scoring

In the third workflow phase, we need to compute the confidence score for axiom candidates, which involves computing the F-measure for each candidate. For Example 1, computing the F-measure means that we need to count the number of instances which belong to `SoccerPlayer`, the number of instances of `Person`$\sqcap$ $\exists$`team.SoccerClub`, as well as the number of instances belonging to both, i.e. `SoccerPlayer` $\sqcap$ `Person` $\sqcap$ $\exists$`team.SoccerClub`. As explained above, the latter

value is divided on the one hand by the first value to obtain the recall, and on the other hand by the second value to get the precision, both resulting in a total score using standard F-measure. For our running example, assume that the following facts about some famous soccer players are given:

SoccerPlayer(Wayne_Rooney)
SoccerPlayer(Lionel_Messi)
Person(Wayne_Rooney)
Person(Lionel_Messi)
Person(Cristiano_Ronaldo)
SoccerClub(FC_Barcelona)
SoccerClub(Manchester_United_F.C.)
SoccerClub(Real_Madrid_C.F.)
team(Wayne_Rooney, Manchester_United_F.C.)
team(Lionel_Messi, FC_Barcelona)
team(Cristiano_Ronaldo, Real_Madrid_C.F.)

In the above example, we would obtain a recall of 100% (2 out of 2) and a precision of 66,7% (2 out of 3), resulting in a total F1 score of 80% for the pattern instantiation $\texttt{SoccerPlayer} \equiv \texttt{Person} \sqcap \exists \texttt{team}.\texttt{SoccerClub}$ of Example 1.

A disadvantage of using this straightforward method of obtaining a score is that it does not take the *support* for an axiom in the knowledge base into account. Specifically, there would be no difference between having 100 out of 100 correct observations or 3 out of 3 correct observations when computing precision and recall.

For this reason, we do not just consider the count, but the average of the 95% confidence interval of the count. This confidence interval can be computed efficiently by using the improved Wald method defined in [1]. Assume we have m observations out of which s were successful, then the approximation of the 95% confidence interval is as follows:

$$\max(0, p' - 1.96 \cdot \sqrt{\frac{p' \cdot (1-p')}{m+4}}) \text{ to } \min(1, p' + 1.96 \cdot \sqrt{\frac{p' \cdot (1-p')}{m+4}})$$

$$\text{with } p' = \frac{s+2}{m+4}$$

This formula is easy to compute and has been shown to be accurate in [1]. In the above case, this would change the precision to 57.3% (previously 66,7%) and the recall to 64.5% (previously 100%), thus leading to a total score of 60.7%. This indicates that there is not much support for the axiom in the knowledge base. However, when larger amounts of data are available, the score would increase and ultimately converge to standard F-score.

6 Optimizations

The disadvantage of the retrieval method in Section 4 is that it performs a *remote count* putting high computational load on the SPARQL endpoint in case of complex axiom patterns and very large data sets. An alternative option is to compute a *relevant fragment* of the knowledge base in a first step and then use that fragment to generate axiom candidates. We generate the relevant fragment as follows: Since we always had named classes on the left hand side of an axiom pattern, we iterate over all classes in the knowledge base. For each class A and axiom pattern $\mathfrak{p}$, we retrieve Concise Bounded Descriptions [8] of depth n for instances of A in a given time limit, where n is the modal depth of $\mathfrak{p}$. This is done via SPARQL CONSTRUCT queries and the result is loaded into a local triple store. We can then query this local store to obtain a local approximation of the recall value for specific instantiations of the axiom pattern. Each instantiation which is above a configurable threshold can then be processed by using the remote count method described in Section 4. In summary, this method performs a local filtering of axiom suggestions and only for viable candidates the exact precision and recall values are computed. The effect of this optimisation is that we reduce the query load on the triple store, in particular several simple queries are send instead of few very expensive queries, which could cause timeouts or overburden endpoints.

7 Experimental Setup

Pattern Frequency Detection. There exists a number of well-known ontology repositories which are frequently used for empirical experimentation. For the pattern frequency detection step, we used the following repositories (details are listed in Table 1):

NCBO BioPortal[9]**,** an open repository of biomedical ontologies [28] that allows users to browse, search and visualize ontologies as well as to annotate and create mappings for ontologies. As of May 2013, the repository contains 385 ontologies in various ontology formats. Due to its ontologies ranging widely in size and complexity, the BioPortal has become a popular corpus for testing OWL ontology applications in recent years, such as pattern analysis [25] and ontology modularization [32].

TONES[10]**,** a curated ontology repository which was developed as part of the TONES project as a means of gathering suitable ontologies for testing OWL applications. It contains 219 well-known test and in-use ontologies, varying strongly in size (up to over 100,000 logical axioms) and complexity (from $\mathcal{EL}$++ to $\mathcal{SROIQ}$). The TONES ontologies are frequently used for empirical studies, such as the prediction of reasoning performance [15] and ontology debugging [19].

[8] CBD: `http://www.w3.org/Submission/CBD/`

[9] `http://bioportal.bioontology.org/`

[10] `http://owl.cs.manchester.ac.uk/repository/`

Table 1. Overview about the ontology repositories used in the experiments

Repository	#Ontologies		#Axioms							
	Total	Error	Total		Tbox		RBox		Abox	
			Avg	Max	Avg	Max	Avg	Max	Avg	Max
TONES	219	12	14,299	1,235,392	8297	658,449	20	932	5981	1,156,468
BioPortal	385	101	25,541	847,755	23,353	847,755	35	1339	2152	220,948
Oxford	793	0	49,997	2,492,761	15,384	2,259,770	25	1365	34,587	2,452,737

Oxford Ontology Library[11]**,** a collection of OWL ontologies which was, similar to the TONES repository, gathered for the purpose of testing OWL tools. The library which was established in late 2012 and currently contains 793 ontologies from 24 different sources, including an existing test corpus and several well-known in-use and test ontologies, the largest containing more than 2,000,000 axioms.

From the selected repositories, we used all ontologies which were available online and could be parsed by the OWL API[12], leading to 1392 ontologies containing approximately 20.5 million terminological axioms. From the ontologies that could not be processed (error column in Table 1), it was either not possible to load them from the given URL (TONES), they could not be parsed by the OWL API (TONES), or they were not publicly accessible (BioPortal).

Pattern Application. For the evaluation of the pattern application, we used 100 randomly chosen classes with at least 5 instances of the well-known DBpedia (`http://dbpedia.org/sparql`) knowledge base, which is a crowd-sourced community effort to extract structured information from Wikipedia. In the used version (3.8), it contains facts about 3.77 million resources, many of them described by the 359 classes, 800 object properties and 859 datatype properties of the DBpedia ontology. We applied the optimization described in Section 6 with a time limit of 60 seconds for the fragment extraction process using thresholds of 0.6. From the results we showed at most 100 pattern instantiations per pattern to 3 non-author evaluators.

8 Results and Discussion

Pattern Frequency Detection. As a result of the pattern frequency detection, we obtained an ordered list of the 15 most frequent non-trivial[13] TBox axiom patterns existing in at least 5 ontologies, as shown in Table 2. It shows how often each axiom pattern occurred (frequency), in how many ontologies it was contained, and the rank by frequency in each ontology repository. In addition, we also report the winsorised frequency: In the sorted list of pattern frequencies

[11] `http://www.cs.ox.ac.uk/isg/ontologies/`
[12] `http://owlapi.sourceforge.net/`
[13] $A \sqsubseteq \top$ and $A \sqsubseteq A$ were filtered.

Table 2. Top 15 TBox axiom patterns ordered by frequency with additional information about the rank (if occurred) in each processed repository. Axiom patterns marked with * were omitted for the user evaluation.

	Pattern	Frequency	Winsorized Frequency	#Ontologies	TONES	BioPortal	Oxford
	1. A $\sqsubseteq$ B	10,174,991	5,757,410	1050	2	1	1
	2. A $\sqsubseteq \exists$ p.B	8,199,457	2,450,582	604	1	2	2
	3. A $\sqsubseteq \exists$ p.($\exists$ q.B)	509,963	441,434	24	n/a	n/a	3
	4. A $\equiv$ B $\sqcap \exists$ p.C	361,777	316,420	319	8	4	4
*	5. B $\sqsubseteq \neg$ A	237,897	53,516	417	3	3	9
	6. A $\equiv$ B	104,508	8332	151	13	34	7
*	7. A $\equiv \exists$ p.B	70,040	11,031	139	36	32	8
	8. $\exists$ p.Thing $\sqsubseteq$ A	41,876	34,795	595	6	7	11
	9. A $\sqsubseteq \forall$ p.B	27,556	21,046	266	4	11	19
	10. A $\equiv$ B $\sqcap \exists$ p.C $\sqcap \exists$ q.D	24,277	20,277	196	11	13	13
	11. A $\equiv$ B $\sqcap$ C	16,597	16,597	78	5	20	22
	12. A $\sqsubseteq \exists$ p.(B $\sqcap \exists$ q.C)	12,453	12,161	84	23	18	15
	13. A $\sqsubseteq \exists$ p.{a}	11,816	4342	65	12	22	20
	14. A $\equiv$ B $\sqcap \exists$ p.(C $\sqcap \exists$ q.D)	10,430	10,430	60	39	21	17
*	15. p $\equiv$ q^-	9943	7393	433	17	19	23

for each ontology (without 0-entries), we set all list entries higher than the 95th percentile to the 95th percentile. This reduces the effect of outliers, i.e. axiom patterns scoring very high because of few very large ontologies frequently using them. The axioms marked with a star (*) are already covered by our previous work on learning an large knowledge bases [7]. We will use the remaining 12 patterns for our evaluation.

Fixpoint Analysis. Based on the results of the pattern frequency detection, we performed a fixpoint analysis, i.e. we analysed how the ranking of the most frequent axiom patterns changed and, thus, investigated whether the ranking of the axiom patterns is fluctuating or stable. To do this, we processed ontology-by-ontology in random order and computed the current corresponding frequency ranking for each axiom pattern. The results shown in Figure 2 indicate that the ranking shows only minor changes after 300 ontologies and, hence, our input set of $\approx$ 1400 ontologies is sufficient.

Manual Evaluation Results. Table 3 shows the result of the manual evaluation, which was done by 3 non-author evaluators. For the evaluation, we used a threshold score of 0.6. If more than 100 axioms were generated for a pattern type, we randomly selected 100 entries. This is shown as the sample size in Table 3. For pattern $A \sqsubseteq \forall p.B$, we could not find axioms above the threshold. Thus, we only evaluated the 11 remaining patterns. The last four columns are the result of a

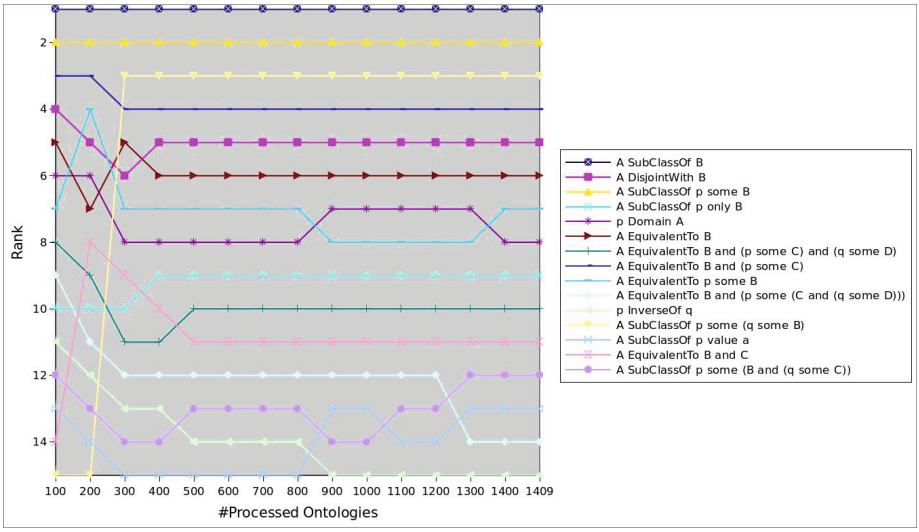


Fig. 2. Top 15 axiom patterns and its sequence of rank when processing the ontologies in random order

manual evaluation. All evaluators independently observed the sample manually and judged the axioms. An advantage of using DBpedia in this context is that the Wikipedia pages provide sufficient background knowledge in most cases in order to judge a particular axiom. Four different categories were used: "correct" indicates that it is likely that they would be accepted by a knowledge engineer, "minor" are axioms which are logically correct, but have modelling problems, "incorrect" are those, which contain conceptual flaws and "not judged" are axioms which could not be evaluated by the authors. Overall, out of 2154 decisions (718 evaluated axioms for each of the 3 reviewers), 48.2% were judged to be correct, 2.7% had minor issues, 49.0% were incorrect and 0 not judged. For a semi-automatic approach with manual validation, this is a reasonable score. The average interrater agreement was substantial, although it was poor for one axiom type. While half of the axiom patterns were frequent in DBpedia, one did not exist at all and 5 were infrequent.

Threshold Analysis. The following diagram shows the correlation between the computed accuracy score of the pattern instantiations and the evaluator judgements, i.e. how many of the pattern instantiations with an accuracy value in a particular interval are correct using majority voting (at least 2 out of 3 reviewers have to judge it as correct).

To perform the analysis, questions were added in 10% buckets by confidence interval (60–70%, > 70% − 80%, > 80% − 90%, > 90% − 100%). Only buckets with at least 5 entries were used (which is why the lines are interrupted). For most axiom types, the trend is that axioms with higher confidence are more likely to be accepted, although two of the 11 axiom types show a decline with higher confidence. The overall trend (dashed line) shows a slope from approx. 50% to

Table 3. Result of the manual evaluation for each axiom pattern

pattern	sample size	manual evaluation in % correct	minor issues	incorrect	κ_{Fleiss}
A ⊑ ∃ p.B	50	88.0	0.7	11.3	24.8
A ⊑ B	47	63.8	2.1	34.0	53.8
A ≡ B	25	10.7	0.0	89.3	44.0
A ≡ ∃ p.B	68	29.9	2.0	68.1	60.4
A ≡ B ⊓ ∃ p.C	100	25.0	3.0	72.0	72.9
A ≡ B ⊓ ∃ p.(C ⊓ ∃ q.D)	100	23.0	5.3	71.7	43.5
A ⊑ ∃ p.(∃ q.B)	71	85.0	3.3	11.7	34.0
A ⊑ ∃ p.(B ⊓ ∃ q.C)	100	87.0	0.3	12.7	-2.8
A ⊑ ∃ p.{a}	15	71.1	0.0	28.9	45.9
A ≡ B ⊓ C	42	14.3	7.1	78.6	46.7
A ≡ B ⊓ ∃ p.C ⊓ ∃ q.D	100	37.0	2.7	59.7	75.0
	718	48.2	2.7	49.0	66.1

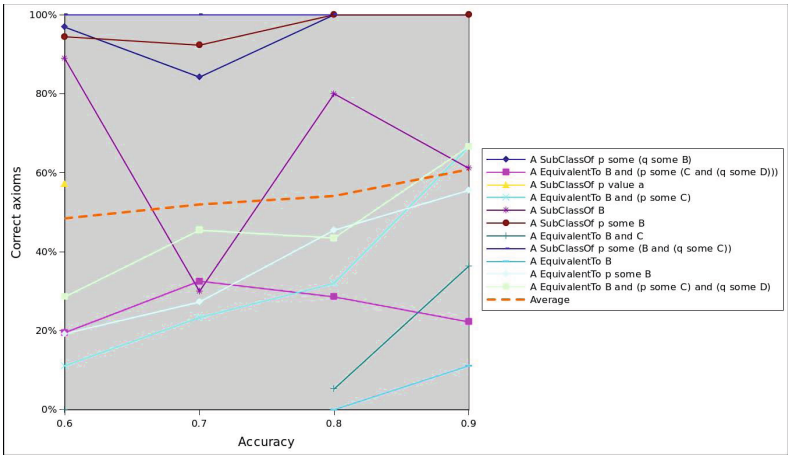


Fig. 3. Correlation between the accuracy value of the pattern instantiations and the confidence of the evaluators

almost 60% indicating that higher accuracy scores result in better axiom suggestions.

Discussion. In this part, we will explain some of the evaluation results and present specific examples. In general, many axioms appeared to be close to human intuition. One of the reasons why axioms were often judged to have "minor issues" is that several suggestions for a particular class and axiom pattern were provided. The lower scoring ones often contained irrelevant parts. An example of

this is $\texttt{GrandPrix} \equiv \texttt{Event} \sqcap (\exists \texttt{poleDriver}.\texttt{Athlete}) \sqcap (\exists \texttt{secondDriver}.\texttt{Agent})$. While logically correct, defining a grand prix via the `Agent` class relation ship of the driver finishing second is not intuitive.

In other cases, there were conceptual flaws, e.g. in the following axioms:

1. $\texttt{Song} \sqsubseteq \exists \texttt{album}.\texttt{MusicalWork}$
2. $\texttt{Song} \sqsubseteq \exists \texttt{artist}.(\exists \texttt{recordLabel}.\texttt{RecordLabel})$
3. $\texttt{BritishRoyalty} \sqsubseteq \exists \texttt{parent}.\texttt{BritishRoyalty}$
4. $\texttt{President} \sqsubseteq \exists \texttt{successor}.\texttt{Person}$
5. $\texttt{SoccerManager} \equiv \texttt{Agent} \sqcap (\exists \texttt{birthPlace}.\texttt{Country}) \sqcap (\exists \texttt{managerClub}.\texttt{SportsTeam})$
6. $\texttt{SoccerClubSeason} \equiv \texttt{Organisation} \sqcap (\exists \texttt{manager}.\texttt{Person}) \sqcap (\exists \texttt{team}.\texttt{SoccerClub})$

The first axiom is not correct, because not each song actually appears on an album, although that is the case for the vast majority of songs in Wikipedia. Similarly, not each song is done by an artist having a record label. The third axiom is flawed, because to our understanding persons can marry British Royals and, e.g. become queen later on, without having been member of the royalty before. The fourth axiom is logically incorrect, because the current president does not have a successor yet. There are many successor relationships in DBpedia, which were suggested by our approach, so this was a major error type in the evaluation. The fifth axiom is also a typical example: The conceptual flaw is that a soccer manager has to manage a soccer team and not just an arbitrary sports team. This suggestion is generated, because soccer data is dominant in Wikipedia relative to other sports. However, in the best suggestion for the class `SoccerManager`, our approach provides the correct axiom. Finally, the last axiom is also incorrect. A soccer club season in DBpedia is modeled as a combination of a soccer club and a specific year, in which the team had a manager. However, `SoccerClubSeason` is a subclass of `Organisation`, which is a modeling error already in DBpedia itself. This particular modeling error had a negative influence on a significant number of axiom suggestions. Overall, the conceptual flaws are either axioms just above the threshold or those for which there is significant statistical evidence for their truth, but corner cases render them invalid. This is also the major reason why we believe that knowledge base construction cannot easily be fully automated.

For equivalent classes, there were 8 axioms above the 60% threshold. However, the DBpedia ontology does not contain classes, which could be seen as equivalent, so all axiom suggestions by our algorithm were classes suggested to be equivalent to their super classes due to having almost the same instances.

9 Related Work

Ontology Enrichment usually involves applying heuristics or machine learning techniques to find axioms, which can be added to an existing ontology. Naturally, different techniques have been applied depending on the specific type of axiom. One of the most complex tasks in ontology enrichment is to find *definitions* of classes. This is strongly related to Inductive Logic Programming (ILP) [27] and

more specifically supervised learning in description logics. Early techniques [9] using *least common subsumers* were later enriched with refinement operators [14]. However, those algorithms tend to produce very long and hard-to-understand class expressions. The algorithms implemented in DL-Learner [22] overcome this problem and investigate the learning problem and the use of top down refinement in detail. However, they require the ontology to be stored in an OWL reasoner in contrast to the work proposed in this article. DL-FOIL [10] is a similar approach, which is based on a mixture of upward and downward refinement of class expressions. Most recently, [18] implements appropriate heuristics and adaptations for learning definitions in ontologies.

A different approach to learning the definition of a named class is to compute the so called *most specific concept* (msc) for all instances of the class. The most specific concept of an individual is the most specific class expression, such that the individual is instance of the expression. One can then compute the *least common subsumer* (lcs) [4] of those expressions to obtain a description of the named class. However, in expressive description logics, an msc does not need to exist and the lcs is simply the disjunction of all expressions. Other approaches, e.g. [23] focus on learning in hybrid knowledge bases combining ontologies and *rules*.

Another enrichment task is *knowledge base completion*. The goal of such a task is to make the knowledge base complete in a particular well-defined sense. For instance, a goal could be to ensure that all subclass relationships between named classes can be inferred. The line of work starting in [29] and further pursued in e.g. [3] investigates the use of *formal concept analysis* for completing knowledge bases. [34] proposes to improve knowledge bases through relational exploration and implemented it in the *RELExO framework*[14]. It focuses on simple relationships and the knowledge engineer is asked a series of questions. The knowledge engineer either must positively answer the question or provide a counterexample.

[35] focuses on learning *disjointness* between classes in an ontology to allow for more powerful reasoning and consistency checking. To achieve this, it can use the ontology itself, but also texts, e.g. Wikipedia articles corresponding to a concept. The article includes an extensive study, which shows that proper modelling disjointness is actually a difficult task, which can be simplified via this ontology enrichment method.

There are further more light-weight ontology enrichment methods. For instance, *taxonomies* can be learned from simple tag structures via heuristics [7,33,31]. All of those approaches follow similar goals. [7] is the base of this article and follows the approach described in Section 2. [33] uses association rule mining with a different set of supported axioms. Learning in this settings is a batch process, which involves building transaction tables and measuring extensional overlap between classes. Finally, [31] follows similar idea, but is restricted to learning property domains and ranges as well as class disjointness. The approach is applied to inconsistency checking in DBpedia.

[14] `http://code.google.com/p/relexo/`

Table 4. Work in ontology enrichment grouped by type or aim of learned structures

Type/Aim	References
Taxonomies	[36,7,33]
Definitions	often done via ILP approaches such as [21,22,18,10,5], genetic approaches [16] have also been used
Super Class Axioms	[18,33,7]
Rules in Ontologies	[23,24]
Disjointness	[35,7,31]
Properties of Properties	[7,11,31]
Completion	formal concept analysis and relational exploration [3,34,30]

Ontology Patterns. There has been a significant amount of research on ontology design patterns with regular workshops on that topic. In particular, we want to refer to [13] for a systematic review on ontology design patterns articles in the Semantic Web community and [12] for a general introduction to the topic. Many patterns are listed at `http://ontologydesignpatterns.org`. Initially, we planned to use this as axiom pattern library. However, it turned out that only a small percentage of the patterns are applicable in the context of knowledge base enrichment and it is difficult to judge their relevancy. Therefore, we decided to perform the described bottom up approach involving several repositories and hundreds of ontologies. In the context of ontology learning, design patterns have been employed in [6]. However, the focus in that scenario is on developing ontologies from textual input, whereas our approach focuses on creating or refining schema structures from existing instance data.

10 Conclusions and Future Work

We presented an approach, which allows to detect frequent axiom usage patterns using $\approx$ 1400 ontologies and converted them into SPARQL query patterns allowing to find those patterns in instance data. This allows to improve knowledge base schemata semi-automatically and is the first scalable schema construction approach based on actual usage patterns to the best of our knowledge. Moreover, it improves the co-evolution of schema and data as well as querying, constraint checking and inference. The evaluation shows that the approach is feasible and able to provide useful suggestions. Nevertheless, we also pointed out corner cases, which are difficult to handle for such a statistical analysis and require human attention. In combination with previous efforts [7,18], we have build an efficient freely available tool, which is able to suggest both TBox and RBox axioms on large knowledge bases accessible via SPARQL endpoints.

Acknowledgement. This work was supported by grants from the European Union's 7th Framework Programme provided for the projects GeoKnow (GA no. 318159) and LOD2 (GA no. 257943).

References

1. Agresti, A., Coull, B.A.: Approximate is better than "exact" for interval estimation of binomial proportions. The American Statistician 52(2), 119–126 (1998)

2. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F. (eds.): The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press (2003)
3. Baader, F., Ganter, B., Sattler, U., Sertkaya, B.: Completing description logic knowledge bases using formal concept analysis. In: IJCAI 2007. AAAI Press (2007)
4. Baader, F., Sertkaya, B., Turhan, A.-Y.: Computing the least common subsumer w.r.t. a background terminology. J. Applied Logic 5(3), 392–420 (2007)
5. Badea, L., Nienhuys-Cheng, S.-H.: A refinement operator for description logics. In: Cussens, J., Frisch, A.M. (eds.) ILP 2000. LNCS (LNAI), vol. 1866, pp. 40–59. Springer, Heidelberg (2000)
6. Blomqvist, E.: Ontocase-automatic ontology enrichment based on ontology design patterns. In: Bernstein, A., Karger, D.R., Heath, T., Feigenbaum, L., Maynard, D., Motta, E., Thirunarayan, K. (eds.) ISWC 2009. LNCS, vol. 5823, pp. 65–80. Springer, Heidelberg (2009)
7. Bühmann, L., Lehmann, J.: Universal OWL axiom enrichment for large knowledge bases. In: ten Teije, A., Völker, J., Handschuh, S., Stuckenschmidt, H., d'Acquin, M., Nikolov, A., Aussenac-Gilles, N., Hernandez, N. (eds.) EKAW 2012. LNCS, vol. 7603, pp. 57–71. Springer, Heidelberg (2012)
8. Bühmann, L., Lehmann, J.: OWL class expression to SPARQL rewriting. Technical report, University of Leipzig (2013), `http://svn.aksw.org/papers/2013/OWL_SPARQL/public.pdf`
9. Cohen, W.W., Borgida, A., Hirsh, H.: Computing least common subsumers in description logics. In: AAAI 1992, pp. 754–760 (1992)
10. Fanizzi, N., d'Amato, C., Esposito, F.: DL-FOIL concept learning in description logics. In: Železný, F., Lavrač, N. (eds.) ILP 2008. LNCS (LNAI), vol. 5194, pp. 107–121. Springer, Heidelberg (2008)
11. Fleischhacker, D., Völker, J., Stuckenschmidt, H.: Mining rdf data for property axioms. In: Meersman, R., Panetto, H., Dillon, T., Rinderle-Ma, S., Dadam, P., Zhou, X., Pearson, S., Ferscha, A., Bergamaschi, S., Cruz, I.F. (eds.) OTM 2012, Part II. LNCS, vol. 7566, pp. 718–735. Springer, Heidelberg (2012)
12. Gangemi, A., Presutti, V.: Ontology design patterns. In: Handbook on Ontologies, pp. 221–243. Springer (2009)
13. Hammar, K., Sandkuhl, K.: The state of ontology pattern research: a systematic review of iswc, eswc and aswc 2005–2009. In: Workshop on Ontology Patterns: Papers and Patterns from the ISWC Workshop, pp. 5–17 (2010)
14. Iannone, L., Palmisano, I., Fanizzi, N.: An algorithm based on counterfactuals for concept learning in the semantic web. Applied Intelligence 26(2), 139–159 (2007)
15. Kang, Y.-B., Li, Y.-F., Krishnaswamy, S.: Predicting reasoning performance using ontology metrics. In: Cudré-Mauroux, P., Heflin, J., Sirin, E., Tudorache, T., Euzenat, J., Hauswirth, M., Parreira, J.X., Hendler, J., Schreiber, G., Bernstein, A., Blomqvist, E. (eds.) ISWC 2012, Part I. LNCS, vol. 7649, pp. 198–214. Springer, Heidelberg (2012)
16. Lehmann, J.: Hybrid learning of ontology classes. In: Perner, P. (ed.) MLDM 2007. LNCS (LNAI), vol. 4571, pp. 883–898. Springer, Heidelberg (2007)
17. Lehmann, J.: DL-Learner: learning concepts in description logics. Journal of Machine Learning Research (JMLR) 10, 2639–2642 (2009)
18. Lehmann, J., Auer, S., Bühmann, L., Tramp, S.: Class expression learning for ontology engineering. Journal of Web Semantics 9, 71–81 (2011)
19. Lehmann, J., Bühmann, L.: ORE - a tool for repairing and enriching knowledge bases. In: Patel-Schneider, P.F., Pan, Y., Hitzler, P., Mika, P., Zhang, L., Pan, J.Z., Horrocks, I., Glimm, B. (eds.) ISWC 2010, Part II. LNCS, vol. 6497, pp. 177–193. Springer, Heidelberg (2010)

20. Lehmann, J., Hitzler, P.: Foundations of refinement operators for description logics. In: Blockeel, H., Ramon, J., Shavlik, J., Tadepalli, P. (eds.) ILP 2007. LNCS (LNAI), vol. 4894, pp. 161–174. Springer, Heidelberg (2008)
21. Lehmann, J., Hitzler, P.: A refinement operator based learning algorithm for the $\mathcal{ALC}$ description logic. In: Blockeel, H., Ramon, J., Shavlik, J., Tadepalli, P. (eds.) ILP 2007. LNCS (LNAI), vol. 4894, pp. 147–160. Springer, Heidelberg (2008)
22. Lehmann, J., Hitzler, P.: Concept learning in description logics using refinement operators. Machine Learning Journal 78(1-2), 203–250 (2010)
23. Lisi, F.A.: Building rules on top of ontologies for the semantic web with inductive logic programming. Theory and Practice of Logic Programming 8(3), 271–300 (2008)
24. Lisi, F.A., Esposito, F.: Learning SHIQ+log rules for ontology evolution. In: SWAP 2008. CEUR Workshop Proceedings, vol. 426. CEUR-WS.org (2008)
25. Mikroyannidi, E., Manaf, N.A.A., Iannone, L., Stevens, R.: Analysing syntactic regularities in ontologies. In: Klinov, P., Horridge, M. (eds.) OWLED. CEUR Workshop Proceedings, vol. 849. CEUR-WS.org (2012)
26. Morsey, M., Lehmann, J., Auer, S., Stadler, C., Hellmann, S.: DBpedia and the Live Extraction of Structured Data from Wikipedia. Program: Electronic Library and Information Systems 46, 27 (2012)
27. Nienhuys-Cheng, S.-H., de Wolf, R.: Foundations of Inductive Logic Programming. LNCS, vol. 1228. Springer, Heidelberg (1997)
28. Rubin, D.L., Moreira, D.A., Kanjamala, P., Musen, M.A.: Bioportal: A web portal to biomedical ontologies. In: AAAI Spring Symposium: Symbiotic Relationships between Semantic Web and Knowledge Engineering, pp. 74–77. AAAI (2008)
29. Rudolph, S.: Exploring relational structures via FLE. In: Wolff, K.E., Pfeiffer, H.D., Delugach, H.S. (eds.) ICCS 2004. LNCS (LNAI), vol. 3127, pp. 196–212. Springer, Heidelberg (2004)
30. Sertkaya, B.: OntocomP system description. In: Grau, B.C., Horrocks, I., Motik, B., Sattler, U. (eds.) Proceedings of the 22nd International Workshop on Description Logics (DL 2009), Oxford, UK, July 27-30. CEUR Workshop Proceedings, vol. 477, CEUR-WS.org (2009)
31. Töpper, G., Knuth, M., Sack, H.: Dbpedia ontology enrichment for inconsistency detection. In: Proceedings of the 8th International Conference on Semantic Systems, pp. 33–40. ACM (2012)
32. Del Vescovo, C., Gessler, D.D.G., Klinov, P., Parsia, B., Sattler, U., Schneider, T., Winget, A.: Decomposition and modular structure of bioportal ontologies. In: Aroyo, L., Welty, C., Alani, H., Taylor, J., Bernstein, A., Kagal, L., Noy, N., Blomqvist, E. (eds.) ISWC 2011, Part I. LNCS, vol. 7031, pp. 130–145. Springer, Heidelberg (2011)
33. Völker, J., Niepert, M.: Statistical schema induction. In: Antoniou, G., Grobelnik, M., Simperl, E., Parsia, B., Plexousakis, D., De Leenheer, P., Pan, J. (eds.) ESWC 2011, Part I. LNCS, vol. 6643, pp. 124–138. Springer, Heidelberg (2011)
34. Völker, J., Rudolph, S.: Fostering web intelligence by semi-automatic OWL ontology refinement. In: Web Intelligence, pp. 454–460. IEEE (2008)
35. Völker, J., Vrandečić, D., Sure, Y., Hotho, A.: Learning disjointness. In: Franconi, E., Kifer, M., May, W. (eds.) ESWC 2007. LNCS, vol. 4519, pp. 175–189. Springer, Heidelberg (2007)
36. Wu, H., Zubair, M., Maly, K.: Harvesting social knowledge from folksonomies. In: Proceedings of the Seventeenth Conference on Hypertext and Hypermedia, HYPERTEXT 2006, pp. 111–114. ACM, New York (2006)

Controlled Query Evaluation over OWL 2 RL Ontologies*

Bernardo Cuenca Grau[1], Evgeny Kharlamov[1], Egor V. Kostylev[2], and Dmitriy Zheleznyakov[1]

[1] Department of Computer Science, University of Oxford
[2] School of Informatics, University of Edinburgh

Abstract. We study confidentiality enforcement in ontology-based information systems where ontologies are expressed in OWL 2 RL, a profile of OWL 2 that is becoming increasingly popular in Semantic Web applications. We formalise a natural adaptation of the Controlled Query Evaluation (CQE) framework to ontologies. Our goal is to provide CQE algorithms that (i) ensure confidentiality of sensitive information; (ii) are efficiently implementable by means of RDF triple store technologies; and (iii) ensure maximality of the answers returned by the system to user queries (thus restricting access to information as little as possible). We formally show that these requirements are in conflict and cannot be satisfied without imposing restrictions on ontologies. We propose a fragment of OWL 2 RL for which all three requirements can be satisfied. For the identified fragment, we design a CQE algorithm that has the same computational complexity as standard query answering and can be implemented by relying on state-of-the-art triple stores.

1 Introduction

Preserving confidentiality of information (i.e., ensuring that sensitive data is only accessible to authorised users) is a critical requirement for the design of information systems. In recent years, Semantic Web technologies have become widespread in many application domains. There is consequently a pressing need for suitable confidentiality enforcement infrastructure in ontology-based information systems which rely on RDF as a data model, SPARQL as a query language, and OWL 2 as a language for describing background knowledge.

In traditional database management systems, confidentiality is enforced by means of mandatory and discretionary access control mechanisms, where access to data items such as tuples, entire relational tables, or database views is granted only to certain (groups of) users. Such access control mechanisms are, however, problematic for ontology-based information systems: explicitly represented RDF

* This work was partially supported by the EU project Optique (FP7-IP-318338), EPSRC project Score!, ERC FP7 grant Webdam (n. 226513), and UK EPSRC project SOCIAM (grant EP/J017728/1). Bernardo Cuenca Grau is also supported by a Royal Society University Research Fellowship.

H. Alani et al. (Eds.): ISWC 2013, Part I, LNCS 8218, pp. 49–65, 2013.

data is assumed to be incomplete and hence new, implicit, data can be derived from the axioms in the ontology via logical reasoning. By granting access to a certain set of RDF triples, system administrators are de facto disclosing a much larger set of implicit triples, some of which a user might not be allowed to know. In contrast, traditional databases are complete, and hence system data is always explicit; as a result, managing access rights is conceptually a simpler problem.

Controlled Query Evaluation (CQE). [1,2,3,4,5,6] is an approach to confidentiality enforcement where system administrators specify in a declarative way the information that cannot be disclosed to users (neither directly nor indirectly via results of previous queries) by means of a *confidentiality policy*. When given a user query, a *censor* checks whether returning the answer would lead to a violation of the corresponding policy and thus to a disclosure of confidential information to unauthorised users; in that case, the censor returns a distorted answer.

CQE in databases is a long standing research area [1,2,4,6,3]; existing work, however, focuses mostly on complete relational databases. CQE for incomplete databases, which are more closely related to ontologies, remains relatively unexplored and research has so far been limited to foundational aspects [5].

In this paper, we are interested in ensuring confidentiality in ontology-based information systems where the relevant ontologies are expressed in the OWL 2 RL profile [7]—a fragment of OWL 2 for which query answering is known to be theoretically tractable in the size of both ontology and data, and efficiently implementable by means of rule-based triple store technologies. OWL 2 RL has become increasingly popular, and state-of-the-art RL reasoners such as OWLim [8] and Oracle's RDF Semantic Graph [9] provide robust and scalable support for SPARQL queries over OWL 2 RL ontologies and RDF data.

Motivated by the CQE paradigm, we study confidentiality enforcement in the scenario described next. We assume that the information in the system consists of background knowledge formalised as an OWL 2 RL ontology, and dataset formalised as a set of unary and binary facts. The ontology is assumed to be fully known to all users (a worst-case situation for confidentiality enforcement), whereas data is assumed to be hidden. Interaction with the system is restricted to a query interface, which allows users to formulate arbitrary conjunctive queries (which constitute the core of SPARQL). A confidentiality policy is represented as a set of facts logically entailed by the ontology and dataset. Given a user query, the system returns a subset of the certain answers to this query over the ontology and dataset determined by the censor. Thus, we adopt the basic case of the CQE paradigm where the censor only filters out problematic answers.

In this scenario, there is a tradeoff between confidentiality and accessibility of information: a censor that returns the empty answer for each query makes the system secure, but also equally useless. Thus, we are interested in *optimal* censors, i.e., those that return maximal sets of answers to queries which still preserve the required confidentiality. Also, CQE can be computationally expensive and to the best of our knowledge practicable algorithms are yet to be developed. We are consequently interested in censors that can be *efficiently implemented*, ideally by relying on the same technology used for query answering in OWL 2 RL.

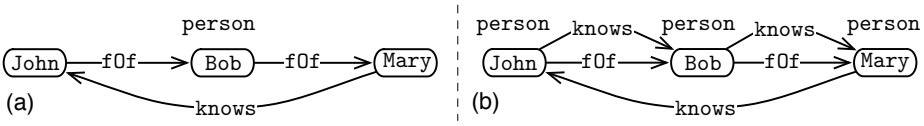


Fig. 1. Dataset $\mathcal{D}_{ex}$ (a), and the same dataset extended with implicit information (b)

The contributions of this paper are as follows. In Section 3 we take existing work on CQE for incomplete databases as a starting point, and present a CQE framework that takes into account the specific features of standard ontology and query languages. In Section 4 we propose the class of *view-definable* censors, which can be implemented by delegating the censor's main computational workload to an OWL 2 RL query answering engine. Roughly speaking, the behaviour of such a censor is determined by what we call a *view*: a dataset that "encodes" the information in the system relevant to the censor's output for any user query. We next explore in Section 5 the formal limitations of our approach and show that censors that are both optimal and view-definable may not exist; furthermore, even if such a censor exists, the corresponding view might be exponentially larger than the system's dataset, thus making efficient implementations difficult. In Section 6, we identify a fragment of OWL 2 RL for which these limitations can be circumvented. Our fragment is able to capture non-trivial extensions of RDF-Schema and is thus relevant for many Semantic Web applications. We consequently provide a practical CQE evaluation algorithm that guarantees both optimality and efficiency if the system's ontology belongs to our fragment. Finally, in Section 7 we observe that there are cases where different optimal view-definable censors exist, and where there is no good reason for choosing one over the others; hence, we study how to deal with such cases.

2 Preliminaries

We assume that all our definitions are parameterised by a first-order signature Σ consisting only of constants, unary predicates, and binary predicates. We also assume first-order logic with equality over Σ, and denote with $\approx$ the special binary equality predicate and $\bot$ the special nullary false predicate. A *dataset* is a finite set of ground (equality-free) atoms over Σ.

Example 1. Consider the following dataset $\mathcal{D}_{ex}$, where the predicate `fOf` represents the "friend of" relation:

$$\texttt{person}(\texttt{Bob}), \quad \texttt{knows}(\texttt{Mary}, \texttt{John}), \quad \texttt{fOf}(\texttt{John}, \texttt{Bob}), \quad \texttt{fOf}(\texttt{Bob}, \texttt{Mary}).$$

A graphical representation of $\mathcal{D}_{ex}$ is given in Figure 1(a). ■

Definition 2 (Rule, ontology). *A* rule *r is a first-order sentence of the form $\forall \boldsymbol{x} \forall \boldsymbol{z}. \varphi(\boldsymbol{x}, \boldsymbol{z}) \rightarrow \psi(\boldsymbol{x})$, where $\boldsymbol{x}$ and $\boldsymbol{z}$ are tuples of variables, $\varphi(\boldsymbol{x}, \boldsymbol{z})$ is a*

conjunction of atoms not mentioning $\approx$ *or* $\bot$*, and* $\psi(\boldsymbol{x})$ *is a single atom. The conjunction* $\varphi(\boldsymbol{x}, \boldsymbol{z})$ *is the* body *of* r *and the atom* $\psi(\boldsymbol{x})$ *is the* head *of* r*; quantifiers are often omitted for simplicity. An* ontology *is a finite set of rules.*

We assume first-order semantics of formulae such as rules, and use $\models$ in the standard way as the logical consequence relation.

Example 3. Consider the ontology $\mathcal{O}_{ex}$ consisting of the following rules:

$$\texttt{knows}(x, y) \rightarrow \texttt{person}(x);\ \texttt{knows}(x, y) \rightarrow \texttt{person}(y);\ \texttt{fOf}(x, y) \rightarrow \texttt{knows}(x, y).$$

Intuitively, $\mathcal{O}_{ex}$ says that only people can participate in the relation `knows`, and if two people are friends (i.e., they participate in the `fOf` relation), then they know each other. In Figure 1(b), we depict the dataset $\mathcal{D}_{ex}$ from Example 1 extended with ground atoms that are logically entailed by $\mathcal{O}_{ex} \cup \mathcal{D}_{ex}$. For example, $\mathcal{O}_{ex} \cup \mathcal{D}_{ex} \models \texttt{person}(\texttt{John})$ and $\mathcal{O}_{ex} \cup \mathcal{D}_{ex} \models \texttt{knows}(\texttt{John}, \texttt{Bob})$. ■

OWL 2 RL was designed as "a syntactic subset of OWL 2 which is amenable to implementation using rule-based technologies" [7]. In particular, each OWL 2 RL knowledge base can be normalised as a set of rules. We make two simplifying assumptions w.r.t. the normative specification of OWL 2 RL. First, we ignore datatypes for simplicity; second, we assume that no constants occur in rules. The latter assumption ensures a clean separation between schema knowledge and data. The following definition characterises a class of rules that is sufficient to capture normative OWL 2 RL under our basic assumptions.

Definition 4 (RL ontology). *An ontology* $\mathcal{O}$ *is an* RL ontology *if it can be partitioned as* $\mathcal{O} = \mathcal{O}' \uplus \mathcal{O}''$ *such that the following properties hold.*

1. *Each rule* r *from* $\mathcal{O}'$ *is constant-free; furthermore, the variables in* r *consist of a single* root variable x *and a set of* branch variables $\boldsymbol{y}$ *such that:*
   (a) *each binary atom in the body of* r *must mention* x *once and only once;*
   (b) *each branch variable* y *occurs in exactly one binary atom in the body;*
   (c) *if the head atom is binary then it is of the form* $y \approx y'$*, for some* y, y' *from* $\boldsymbol{y}$*, and the binary atoms in the body, mentioning* y *and* y'*, use the same predicate symbol and have* y *and* y' *on the same position.*
2. *Each rule in* $\mathcal{O}''$ *is of one of the following forms:*
   (a) $R(x, y) \rightarrow S(x, y)$*; or*
   (b) $R(x, y) \wedge S(y, z) \rightarrow T(x, z)$*; or*
   (c) $R(x, y) \rightarrow S(y, x)$*; or*
   (d) $R(x, y) \wedge S(x, y) \rightarrow \bot$.

The OWL 2 RL specification requires certain *global restrictions* to hold in order to ensure that OWL 2 RL is a syntactic fragment of OWL 2 DL (e.g., transitive properties cannot occur in cardinality constraints) [7,10]. Such restrictions are not reflected in Definition 4 since they are immaterial to our results.

Example 5. The ontology $\mathcal{O}_{ex}$ is an RL ontology. Moreover, for $\mathcal{O}_{ex}$, $\mathcal{O}'_{ex}$ consists of the first two rules, while $\mathcal{O}''_{ex}$ consists of the last rule only. ■

Definition 6 (Conjunctive query). *A* conjunctive query $Q(\boldsymbol{x})$ *is a first-order formula of the form* $\exists \boldsymbol{y}.\, \varphi(\boldsymbol{x}, \boldsymbol{y})$*, where* $\boldsymbol{x}$ *and* $\boldsymbol{y}$ *are tuples of variables and* $\varphi(\boldsymbol{x}, \boldsymbol{y})$ *is a conjunction of atoms.*

We will write Q instead of $Q(\boldsymbol{x})$ when $\boldsymbol{x}$ is irrelevant or clear from the context.

Definition 7 (Query answering). *Let* $\mathcal{O}$ *be an ontology,* $\mathcal{D}$ *be a dataset, and let* $Q(\boldsymbol{x})$ *be a conjunctive query. A tuple* $\boldsymbol{t}$ *of constants is a* certain answer *to* $Q(\boldsymbol{x})$ *w.r.t.* $\mathcal{O}$ *and* $\mathcal{D}$ *if* $\mathcal{O} \cup \mathcal{D} \models Q(\boldsymbol{t})$*. The set of all certain answers to a conjunctive query* $Q(\boldsymbol{x})$ *w.r.t.* $\mathcal{O}$ *and* $\mathcal{D}$ *is denoted by* $\mathsf{cert}(Q, \mathcal{O}, \mathcal{D})$*.*

Example 8. Consider the following queries:

$$Q_1(x) = \texttt{person}(x); \qquad Q_2(x) = \exists y, z.\, \texttt{fOf}(x, y) \wedge \texttt{fOf}(y, z) \wedge \texttt{knows}(z, x).$$

Clearly, $\mathsf{cert}(Q_1, \mathcal{O}_{ex}, \mathcal{D}_{ex}) = \{\texttt{John}, \texttt{Bob}, \texttt{Mary}\}$ since all constants in $\mathcal{D}_{ex}$ are entailed to be persons (c.f., Figure 1(b)). Also, $\mathsf{cert}(Q_2, \mathcal{O}_{ex}, \mathcal{D}_{ex}) = \{\texttt{John}\}$ since `John` is a friend of `Bob`, `Bob` is a friend of `Mary`, and `Mary` knows `John`.

3 Controlled Query Evaluation for Ontologies

Our approach to confidentiality enforcement in ontology-based information systems was inspired by the CQE paradigm for incomplete databases first proposed by Biskup and Weibert [5], which we briefly describe next.

A CQE system in [5] stores a database and a *policy*, which are both defined as sets of propositional sentences. The policy is under the control of the system administrators, and its goal is to declaratively specify the information that is to be kept secret. Both database and policy are hidden from users, and interaction with the system is limited to a query interface. Given a (propositional) user query the system does not directly return the correct answer; instead, a *censor* decides whether the answer needs to be modified according to the policy.

Let $q_1, \ldots, q_n$ be any finite sequence of such user queries, let $v_1, \ldots, v_n$ be the answers (truth values) to these queries returned by the censor for the system's database $\mathbf{D}$, and let γ be an arbitrary sentence in the policy. The *confidentiality* of γ is compromised if the truth values $v_1, \ldots, v_n$ fully determine the truth value of γ over $\mathbf{D}$. In other words, to preserve confidentiality of γ there must exist some other database $\mathbf{D}'$ such that the censor evaluates the queries $q_1, \ldots, q_n$ to the same values $v_1, \ldots, v_n$ over $\mathbf{D}'$, but γ does not hold in $\mathbf{D}'$. This means that $\mathbf{D}$ and $\mathbf{D}'$ are indistinguishable w.r.t. the user queries and hence the user cannot decide whether γ holds or not. Hereby, the censor must preserve confidentiality of all the sensitive data in the policy—that is, it should make sure that users cannot derive any sentence in the policy by posing any finite set of queries.

Our framework focuses on ontology-based information systems and hence deviates from [5] in order to better reflect the specific features of standard ontology and query languages. In the remainder of this section, we formally describe the elements of our approach.

3.1 Policies and CQE-Instances

We start with some assumptions made in our framework. Following [5], we assume that both dataset and policy are hidden and that users can pose arbitrary (conjunctive) queries to a query interface. We will assume, however, that the system's ontology is fully known to all users. The rationale behind this assumption is twofold. On the one hand, information represented in ontologies is typically common knowledge, and it is dangerous to enforce confidentiality by relying on users' unawareness of rather straightforward constraints; on the other hand, public availability of the system's background knowledge is key to improving access to information: familiarity with the rules in the ontology can be invaluable for users to formulate accurate queries. Furthermore, in our setting, information is under the control of system developers and domain experts; thus, inconsistencies have been resolved before the query interface is made available to users. We consequently assume that query answering is always performed over a satisfiable ontology and dataset. This assumption makes query results meaningful to users. Finally, we assume that a policy is represented by a set of ground atoms that are logically entailed by the ontology and dataset in the system.

To sum up, the relevant content of the CQE system for the purpose of confidentiality enforcement is formalised in the following definition.

Definition 9 (Policy, CQE-instance). *Let $\mathcal{O}$ be an ontology and let $\mathcal{D}$ be a dataset such that $\mathcal{O} \cup \mathcal{D}$ is satisfiable. A* policy $\mathcal{P}$ *for $\mathcal{O}$ and $\mathcal{D}$ is a dataset such that $\mathcal{O} \cup \mathcal{D} \models \mathcal{P}$, and a* CQE-instance *is the triple $\mathbf{I} = (\mathcal{O}, \mathcal{D}, \mathcal{P})$.*

Example 10. The dataset $\mathcal{P}_{ex} = \{\texttt{knows}(\texttt{Mary}, \texttt{John})\}$ is a policy for our running example ontology $\mathcal{O}_{ex}$ and dataset $\mathcal{D}_{ex}$ since $\mathcal{O}_{ex} \cup \mathcal{D}_{ex} \models \mathcal{P}_{ex}$. The triple $\mathbf{I}_{ex} = (\mathcal{O}_{ex}, \mathcal{D}_{ex}, \mathcal{P}_{ex})$ thus constitutes a CQE-instance. ■

3.2 Censors and Confidentiality Preservation

In [5], Biskup and Weibert consider censors that can distort query answers in various ways. We adopt a pragmatic approach where censors are required to return a sound, but possibly incomplete set of certain answers. Thus, the goal of such a censor is limited to filtering out answers which may compromise the policy. In contrast to [5], our censors never return unsound answers, or reject queries.

Definition 11 (Censor). *A* censor cens *for a CQE-instance $\mathbf{I} = (\mathcal{O}, \mathcal{D}, \mathcal{P})$ is a function which maps each conjunctive query Q to a subset of $\mathsf{cert}(Q, \mathcal{O}, \mathcal{D})$.*

To align with the semantics of OWL 2, we adopt a notion of confidentiality preservation that is formulated directly in terms of first-order models and entailment. Specifically, with a censor cens for a CQE-instance $\mathbf{I} = (\mathcal{O}, \mathcal{D}, \mathcal{P})$, we associate the following (possibly infinite) set of first-order sentences:

$$\mathcal{F}(\mathsf{cens}) = \{Q(\boldsymbol{t}) \mid \boldsymbol{t} \in \mathsf{cens}(Q), Q(\boldsymbol{x}) \text{ is a conjunctive query}\}.$$

The set $\mathcal{F}(\mathsf{cens})$ intuitively represents all the information that a user can potentially gain by interacting with the query interface. Since a user can ask only a finite (yet unbounded) number of arbitrary queries, the information that the user can gather from the system can be captured by a finite subset of $\mathcal{F}(\mathsf{cens})$. Confidentiality preservation then amounts to ensuring that no finite subset of $\mathcal{F}(\mathsf{cens})$ can logically entail an atom in the policy when coupled with $\mathcal{O}$.

Definition 12 (Confidentiality preservation). *Let* $\mathbf{I} = (\mathcal{O}, \mathcal{D}, \mathcal{P})$ *be a CQE-instance. A censor* cens *for* $\mathbf{I}$ *is* confidentiality preserving *if for each ground atom* α *in the policy* $\mathcal{P}$ *and each finite subset* $\mathcal{F}$ *of* $\mathcal{F}(\mathsf{cens})$ *it holds that* $\mathcal{O} \cup \mathcal{F} \not\models \alpha$.

Definition 12 is consistent with the notion of confidentiality preservation in [5]. Indeed, if a censor cens is confidentiality preserving for $\mathbf{I}$, then a user cannot entail any confidential information from $\mathcal{P}$ regardless of what queries they pose. Thus, for each atom α in the policy $\mathcal{P}$ and each finite subset $\mathcal{F}$ of $\mathcal{F}(\mathsf{cens})$ there is a model of $\mathcal{O} \cup \mathcal{F}$ in which α does not hold. Moreover, since $\mathcal{O}$ is an OWL 2 RL ontology and $\mathcal{F}$ is a finite set of positive existential first-order sentences, there always exists a finite model M, which can be seen as a database instance in the sense of Biskup and Weibert. Since $M \models \mathcal{F}$, the model M cannot be distinguished from $\mathcal{D}$ using the query answers returned by the censor. In contrast, $\mathcal{D}$ and M differ w.r.t. the atom α in the policy, which implies that the user cannot decide whether α holds or not in $\mathcal{D}$ based on returned query answers alone.

3.3 Information Access vs. Confidentiality Tradeoff

There is a tradeoff between confidentiality preservation and accessibility of information. On the one hand, a censor for a CQE-instance that returns the empty set of answers for each query is clearly confidentiality preserving, but it also does not provide any useful information; on the other hand, a censor that returns all the certain answers to each query maximises information accessibility, but may not be confidentiality preserving. Hence, we are interested in *optimal* censors, which distort the answer only if necessary for enforcing confidentiality.

Definition 13 (Optimality). *Let* $\mathbf{I}$ *be a CQE-instance, and let* cens *be a confidentiality preserving censor for* $\mathbf{I}$. *The censor* cens *is* optimal *if no other censor* $\mathsf{cens}' \neq \mathsf{cens}$ *for* $\mathbf{I}$ *exists such that (i)* cens' *is confidentiality preserving; and (ii)* $\mathsf{cens}(Q) \subseteq \mathsf{cens}'(Q)$ *holds for each conjunctive query* Q.

As we will discuss later on, there can be several optimal censors for a given CQE-instance. In general, however, there is no good reason for choosing one over the others, so we will design an algorithm which constructs all of them; however, we will also study situations were a unique optimal censor is guaranteed to exist.

We conclude this section with a useful characterisation of the optimality of cens in terms of its associated theory $\mathcal{F}(\mathsf{cens})$.

Proposition 14. *Let* cens *be a confidentiality preserving censor for a CQE-instance* $\mathbf{I} = (\mathcal{O}, \mathcal{D}, \mathcal{P})$. *Then,* cens *is optimal iff for each conjunctive query* $Q(\boldsymbol{x})$ *and each tuple* $\boldsymbol{t} \in \mathsf{cert}(Q, \mathcal{O}, \mathcal{D})$ *the fact that* $\mathcal{O} \cup \mathcal{F}(\mathsf{cens}) \cup \{Q(\boldsymbol{t})\} \not\models \alpha$ *holds for each* $\alpha \in \mathcal{P}$ *implies that* $\mathcal{O} \cup \mathcal{F}(\mathsf{cens}) \models Q(\boldsymbol{t})$.

4 View-Based Controlled Query Evaluation

As already discussed, the main task of the censor in a typical CQE system is to receive answers to user queries as computed by the query answering engine, and to decide, according to the policy, which answers are safe to return to the user and which ones must be distorted. Such a censor is usually conceived as a separate component, which is implemented on top of the query answering engine. Unsurprisingly, implementing the censor of a CQE system becomes a major challenge. The censor's task can be computationally very expensive, and the cost of the censor's evaluation adds to the cost of query answering. Furthermore, implementing the censor may require dedicated algorithms, and, in particular, the highly optimised infrastructure available for query answering might not be reusable. Hence, performance of a CQE system can be significantly affected by the confidentiality enforcement component, and, as a result, the system might not be practically feasible in performance-critical situations.

To address these challenges, we develop a novel approach that deviates from the mainstream separation of query answering engine and censor as different components of a CQE system. More specifically, we propose to exploit the available OWL 2 RL triple store infrastructure as much as possible, by delegating the censor's main computational workload to the query answering engine.

Our key idea is to associate to each CQE-instance $\mathbf{I} = (\mathcal{O}, \mathcal{D}, \mathcal{P})$ a new dataset, which we call a *view*. Such a view $\mathcal{V}$ determines a censor $\mathsf{cens}_{\mathcal{V}}$ in the sense that, for each input user query Q, the censor's output $\mathsf{cens}_{\mathcal{V}}(Q)$ is uniquely and trivially extractable from the set $\mathsf{cert}(Q, \mathcal{O}, \mathcal{V})$ of all certain answers to Q w.r.t. the ontology $\mathcal{O}$ and the view $\mathcal{V}$. Since the view $\mathcal{V}$ is associated only with $\mathbf{I}$ and is query-indenpendent, it also does not need to be recomputed until the underlying dataset $\mathcal{D}$ is updated. In this way, the main workload of the censor in a typical user session boils down to the computation of certain answers, which can be fully delegated to the query answering engine.

Obviously, if we want the censor $\mathsf{cens}_{\mathcal{V}}$ to enjoy the properties we are after, the view $\mathcal{V}$ must be constructed with care. In order for $\mathsf{cens}_{\mathcal{V}}$ to be indeed a censor, $\mathcal{V}$ must not lead to spurious query answers. Furthermore, in order for $\mathsf{cens}_{\mathcal{V}}$ to be confidentiality preserving, $\mathcal{V}$ and $\mathcal{O}$ should not entail any atom in $\mathcal{P}$. Finally, in order for $\mathsf{cens}_{\mathcal{V}}$ to be optimal, $\mathcal{V}$ must "encode" as much information from $\mathcal{D}$ as possible. We next illustrate these ideas with an example.

Example 15. We construct a view $\mathcal{V}_{ex}$ for our example CQE-instance $\mathbf{I}_{ex} = (\mathcal{O}_{ex}, \mathcal{D}_{ex}, \mathcal{P}_{ex})$. Since the policy $\mathcal{P}_{ex}$ contains the atom $\alpha = \mathtt{knows}(\mathtt{Mary}, \mathtt{John})$, we cannot include α in $\mathcal{V}_{ex}$. An obvious possibility would be to define $\mathcal{V}_{ex}$ as $\mathcal{D}_{ex} \setminus \{\alpha\}$, and then $\mathsf{cens}_{\mathcal{V}_{ex}}$ as the function that returns $\mathsf{cert}(Q, \mathcal{O}_{ex}, \mathcal{V}_{ex})$ for each conjunctive query Q. Clearly, $\mathsf{cens}_{\mathcal{V}_{ex}}$ is a censor for $\mathbf{I}_{ex}$, since $\mathsf{cert}(Q, \mathcal{O}_{ex}, \mathcal{V}_{ex}) \subseteq \mathsf{cert}(Q, \mathcal{O}_{ex}, \mathcal{D}_{ex})$, i.e., $\mathsf{cens}_{\mathcal{V}_{ex}}$ returns only sound answers. Furthermore, $\mathsf{cens}_{\mathcal{V}_{ex}}$ is confidentiality preserving: since $\mathcal{O}_{ex} \cup \mathcal{V}_{ex} \not\models \alpha$ and $\mathcal{O}_{ex} \cup \mathcal{V}_{ex} \models \mathcal{F}(\mathsf{cens}_{\mathcal{V}_{ex}})$, it is clear that $\mathcal{O}_{ex} \cup \mathcal{F}(\mathsf{cens}_{\mathcal{V}_{ex}}) \not\models \alpha$, as required by Definition 12. The censor $\mathsf{cens}_{\mathcal{V}_{ex}}$ is, however, not optimal. To see this, consider the query $Q(x) = \exists y, z.\, \mathtt{fOf}(x, y) \wedge \mathtt{knows}(y, z)$ asking for everyone who is a friend of someone

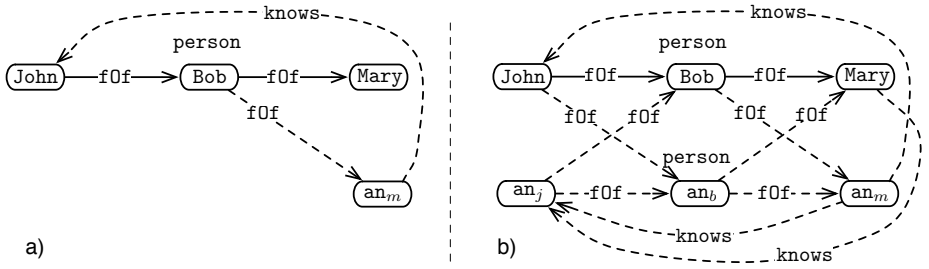


Fig. 2. View $\mathcal{V}_{ex}$ (a), and view defining an optimal censor for $\mathbf{I}_{ex} = (\mathcal{O}_{ex}, \mathcal{D}_{ex}, \mathcal{P}_{ex})$ (b); dashed arrows represent binary atoms that do not occur in $\mathcal{D}_{ex}$

who in turn knows somebody else. We have $\mathsf{cert}(Q, \mathcal{O}_{ex}, \mathcal{D}_{ex}) = \{\mathtt{John}, \mathtt{Bob}\}$, but $\mathsf{cert}(Q, \mathcal{O}_{ex}, \mathcal{V}_{ex}) = \{\mathtt{John}\}$; nevertheless, answering this query correctly is "harmless" in the sense that $\mathcal{O}_{ex} \cup \mathcal{F}(\mathsf{cens}_{\mathcal{V}_{ex}}) \cup \{Q(\mathtt{Bob})\} \not\models \alpha$ (c.f. Proposition 14).

Clearly, we cannot add α back into $\mathcal{V}_{ex}$ without compromising the policy, and hence our only choice is to "encode" the missing information in $\mathcal{V}_{ex}$ by some other means. A possibility is to extend the domain of $\mathcal{V}_{ex}$ with an "anonymised copy" $\mathtt{an}_m$ of $\mathtt{Mary}$, and extend $\mathcal{V}_{ex}$ with the atoms $\mathtt{fOf}(\mathtt{Bob}, \mathtt{an}_m)$ and $\mathtt{knows}(\mathtt{an}_m, \mathtt{John})$ (see Figure 2(a)). As a result, we obtain $\mathsf{cert}(Q, \mathcal{O}_{ex}, \mathcal{V}_{ex}) = \{\mathtt{John}, \mathtt{Bob}\}$ and $\mathcal{O}_{ex} \cup \mathcal{V}_{ex} \not\models \alpha$ as we wanted. There is, however, an undesired effect to this modification: for queries such as $Q'(x) = \exists y.\, \mathtt{knows}(x, y)$ we would obtain the fresh constant $\mathtt{an}_m$ as a spurious answer. The obvious fix is to filter out such answers syntactically, and only return those answers in $\mathsf{cert}(Q, \mathcal{O}_{ex}, \mathcal{V}_{ex})$ that mention only constants from the domain of $\mathcal{D}_{ex}$. Although such extended view is still not optimal, we can reiterate this procedure until we achieve optimality. As we will see, the view depicted in Figure 2(b) defines an optimal censor for $\mathbf{I}_{ex}$. ■

We are ready to define the notion of a view $\mathcal{V}$ and its corresponding censor.

Definition 16 (View). *Let* $\mathbf{I} = (\mathcal{O}, \mathcal{D}, \mathcal{P})$ *be a CQE-instance. A* view *for* $\mathbf{I}$ *is a dataset* $\mathcal{V}$ *which satisfies the following properties:*

(i) $\mathcal{O} \cup \mathcal{V} \not\models \alpha$ *for each* $\alpha \in \mathcal{P}$*; and*
(ii) $\boldsymbol{t} \in \mathsf{cert}(Q, \mathcal{O}, \mathcal{V})$ *implies* $\boldsymbol{t} \in \mathsf{cert}(Q, \mathcal{O}, \mathcal{D})$ *for each conjunctive query* Q *and each tuple* $\boldsymbol{t}$ *of constants from* $\mathcal{D}$*.*

Definition 17 (View-based censor). *Let* $\mathbf{I} = (\mathcal{O}, \mathcal{D}, \mathcal{P})$ *be a CQE-instance and let* $\mathcal{V}$ *be a view for* $\mathbf{I}$*. The* censor based on $\mathcal{V}$ *(or* view-based censor *when* $\mathcal{V}$ *is clear) is the function* $\mathsf{cens}_{\mathcal{V}}$ *mapping a conjunctive query* Q *to the set of tuples*

$$\{\boldsymbol{t} \mid \boldsymbol{t} \in \mathsf{cert}(Q, \mathcal{O}, \mathcal{V}), \boldsymbol{t} \text{ has constants only from } \mathcal{D}\}.$$

By Property *(ii)* in Definition 16, each view-based censor is indeed a censor. Proposition 18 establishes that view-based censors are confidentiality preserving.

Proposition 18. *Let* $\mathbf{I}$ *be a CQE-instance and let* $\mathcal{V}$ *be a view for* $\mathbf{I}$*. Then* $\mathsf{cens}_{\mathcal{V}}$ *is a confidentiality preserving censor for* $\mathbf{I}$*.*

5 Limitations of View-Based Censors

Before investigating the design of efficient view-based CQE algorithms, we first explore the theoretical limitations of our approach. In this section we answer the following important questions about an arbitrary CQE-instance $\mathbf{I}$.

1. Is an optimal view-based censor for $\mathbf{I}$ guaranteed to exist?
2. How large can be the smallest view defining an optimal censor for $\mathbf{I}$?

We answer the first question negatively: the presence of equality in the ontology can preclude the existence of an optimal view-based censor, in the sense that the "view" corresponding to such a censor would be necessarily infinite. As a result, there are CQE-instances for which all view-based censors are not optimal.

Concerning the second question, we show that even if the ontology does not contain equalities, the smallest view associated to an optimal censor can be at least exponentially larger than the given instance. This is a crucial limitation in practice, since such a censor would need to compute certain answers over an exponentially large dataset, with the obvious negative effect on performance.

In Section 6 we will restrict the form of ontology rules to guarantee the existence of optimal censors based on small views.

5.1 Non-existence of Optimal View-Based Censors

We say that a censor cens for a CQE-instance $\mathbf{I}$ is *view-definable* if there exists a view $\mathcal{V}$ for $\mathbf{I}$ such that $\mathsf{cens} = \mathsf{cens}_{\mathcal{V}}$. The following theorem establishes that for some CQE-instances view-definability and optimality of a censor are in conflict.

Theorem 19. *There exists a CQE-instance* $\mathbf{I} = (\mathcal{O}, \mathcal{D}, \mathcal{P})$ *with an* RL *ontology* $\mathcal{O}$*, for which no censor exists that is both view-definable and optimal.*

The intuition behind the proof is given by means of the following example.

Example 20. Consider the CQE-instance $\mathbf{I} = (\mathcal{O}, \mathcal{D}, \mathcal{P})$, where $\mathcal{P} = \{\mathtt{emp}(\mathtt{John})\}$, $\mathcal{D} = \{\mathtt{manages}(\mathtt{John}, \mathtt{John})\}$, and $\mathcal{O}$ is defined as follows:

$$\mathcal{O} = \{\mathtt{manages}(x_1, y) \wedge \mathtt{manages}(x_2, y) \rightarrow x_1 \approx x_2;\ \ \mathtt{manages}(x, y) \rightarrow \mathtt{emp}(y)\}.$$

The rules in $\mathcal{O}$ say that a person can only be managed by a single manager and that everyone who is managed is an employee. Consider also the following (infinite) sequence of conjunctive queries (for $k \geq 1$):

$$Q_k(x) = \exists x_1, \ldots, x_k.\, \mathtt{manages}(x, x_1) \wedge \ldots \wedge \mathtt{manages}(x_{k-1}, x_k).$$

Clearly, $\mathtt{John}$ is the only certain answer to each Q_k w.r.t. $\mathcal{O}$ and $\mathcal{D}$. Furthermore, answering each of these queries correctly is "harmless" for the confidentiality of the policy, and hence each optimal censor for $\mathbf{I}$ must answer these queries correctly. Imagine that such an optimal censor is based on some view $\mathcal{V}$; in order for $\mathtt{John}$ to be returned as an answer to a given Q_k, $\mathcal{V}$ must contain atoms

$\texttt{manages}(\texttt{John}, a_1), \ldots, \texttt{manages}(a_{k-1}, a_k)$. Since k is unbounded and the view must be finite, some of the individuals a_i must be equal; but then, the fact that `manages` is axiomatised in $\mathcal{O}$ as inverse-functional causes the "management chain" in $\mathcal{V}$ to "collapse" into a cycle involving `John`. This compromises the policy since $\mathcal{O} \cup \mathcal{V}$ implies that `John` is managed by someone, and hence is an employee. ■

5.2 Exponential Size of Views Defining Optimal Censors

Consider now the situation where an optimal view-based censor exists for a given instance. The following theorem says that the smallest view associated to any such optimal censor might necessarily be of size at least exponential in the size of the given instance. In what follows, $|\mathcal{O}|$ and $|\mathcal{D}|$ denote the number of atoms in the rules of the ontology $\mathcal{O}$ and in the dataset $\mathcal{D}$, respectively.

Theorem 21. *There exists a sequence of CQE-instances* $\mathbf{I}_n = (\mathcal{O}_n, \mathcal{D}_n, \mathcal{P})$ *for* $n \geq 1$ *such that* $\mathcal{O}_n$ *is an equality-free* RL *ontology,* $|\mathcal{O}_n| \in O(n)$, $|\mathcal{D}_n| \in O(n)$, *and each view* $\mathcal{V}$ *for* $\mathbf{I}_n$ *with* $\mathsf{cens}_{\mathcal{V}}$ *optimal is such that* $|\mathcal{V}| \in \Omega(2^n)$.

Again, we explain the main ideas of the proof by means of an example.

Example 22. Next we give the second element $\mathbf{I}_2 = (\mathcal{O}_2, \mathcal{D}_2, \mathcal{P})$ of the sequence $\mathbf{I}_n$ of CQE-instances. Let $\mathcal{P} = \{\texttt{executive}(\texttt{John})\}$ and

$$\mathcal{O}_2 = \{A_1^i(x) \wedge A_2^i(x) \wedge \texttt{managedBy}(x, y) \rightarrow \texttt{executive}(y) \mid 1 \leq i \leq 2\},$$
$$\mathcal{D}_2 = \{\texttt{managedBy}(\texttt{Bob}, \texttt{John})\} \cup \{A_j^i(\texttt{Bob}) \mid 1 \leq i, j \leq 2\}.$$

The ontology $\mathcal{O}_2$ has two rules with four atoms in each, and the dataset $\mathcal{D}_2$ has $2 \times 2 + 1$ atoms. Consider the following four queries, which ask for those people who manage someone satisfying a given subset of requirements:

$$Q_{j_1,j_2}(y) = \exists x.\, A_{j_1}^1(x) \wedge A_{j_2}^2(x) \wedge \texttt{managedBy}(x, y); \quad 1 \leq j_1, j_2 \leq 2.$$

Clearly, `John` is the only certain answer to each of these queries w.r.t. $\mathcal{O}_2$ and $\mathcal{D}_2$. Answering these queries correctly does not compromise the policy, because none of the pairs of A_j^i from the queries occur together in the body of any rule in $\mathcal{O}_2$.

So, in order to be optimal, a censor $\mathsf{cens}_{\mathcal{V}}$ must answer all these queries correctly and, for this, the view $\mathcal{V}$ must contain four "witnessing" constants a_{j_1,j_2} for each $1 \leq j_1, j_2 \leq 2$, such that $A_{j_1}^1(a_{j_1,j_2})$, $A_{j_2}^2(a_{j_1,j_2})$, and $\texttt{managedBy}(a_{j_1,j_2}, \texttt{John})$ are in $\mathcal{V}$. Furthermore, any pair of these constants cannot be identified into a single one, since otherwise the user would be able to derive the policy atom.

Similarly, for any other $n \geq 1$, the domain of the view has to contain 2^n different constants $a_{j_1,\ldots,j_n}$ (each witnessing a different query $Q_{j_1,\ldots,j_n}$). ■

This example exploits that RL ontologies allow rules in which unary atoms refer to branch variables. Alternatively, a similar example can be constructed by using rules of the form *2(b)* in Definition 4 (also known as *role chain rules*).

6 Efficient View-Based Controlled Query Evaluation

Theorems 19 and 21 show that ensuring optimality comes at the expense of practicality. The proofs of these theorems, however, rely on very specific OWL 2 RL constructs: Theorem 19 critically depends on equality, whereas Theorem 21 requires a rule that mentions a unary atom involving a branch variable (or, alternatively, a rule of the form *2(b)* in Definition 4).

We next present a fragment RL^- of OWL 2 RL for which the limitations from Section 5 can be circumvented. We show that for any CQE-instance involving an RL^- ontology it is possible to construct an optimal view in polynomial time (in the size of $\mathbf{I}$). We start with the definition of RL^-.

Definition 23 (RL^- ontology). *An* RL^- ontology *is an RL ontology* $\mathcal{O} = \mathcal{O}' \uplus \mathcal{O}''$ *satisfying the following restrictions.*

1. *Each rule* r *in* $\mathcal{O}'$ *is equality-free. Furthermore, each unary atom in* r *(both in the head and body) mentions only the root variable of* r*.*
2. *There is no rule of the form 2(b) from Definition 4 in* $\mathcal{O}''$*.*

In particular, this definition ensures that positive rules in $\mathcal{O}'$ are of the form

$$\bigwedge\nolimits_i A_i(x) \wedge \bigwedge\nolimits_j R_j(x, y_j) \wedge \bigwedge\nolimits_k S_k(y_k, x) \rightarrow B(x).$$

The ontologies in Examples 20 and 22 are not RL^- ontologies. Nevertheless, RL^- is powerful enough to capture the rules corresponding to RDFS, including subclass and subproperty axioms (i.e., rules of the form $A(x) \rightarrow B(x)$ and $R(x, y) \rightarrow S(x, y)$) as well as property domain and range axioms (i.e., rules $R(x, y) \rightarrow A(x)$ and $R(x, y) \rightarrow A(y)$). Additionally, RL^- goes well beyond RDFS and can capture other useful kinds of OWL 2 RL rules.

Example 24. Our example ontology $\mathcal{O}_{ex}$ is expressible as RDFS rules and hence also in RL^-. The following rules are RL^-, but with no correspondence in RDFS:

$$\texttt{fOf}(x, y) \rightarrow \texttt{fOf}(y, x); \tag{1}$$

$$\texttt{emp}(x) \wedge \texttt{manages}(x, y) \rightarrow \texttt{manager}(x); \tag{2}$$

$$\texttt{student}(x) \wedge \texttt{onpayroll}(x) \rightarrow \texttt{PHDStudent}(x). \tag{3}$$

Rule (1) axiomatises `fOf` as symmetric. Rule (2) says that employees managing others are managers. Rule (3) says that paid students must be doing a PhD. ■

We next present an algorithm ComputeView (c.f. Algorithm 1) that takes as input a CQE-instance $\mathbf{I} = (\mathcal{O}, \mathcal{D}, \mathcal{P})$ such that $\mathcal{O}$ is an RL^- ontology and returns a view $\mathcal{V}$ for $\mathbf{I}$ such that $\mathsf{cens}_{\mathcal{V}}$ is guaranteed to be an optimal censor. This algorithm works in polynomial time and, hence, computes $\mathcal{V}$ of polynomial size.

The algorithm ComputeView starts by creating an anonymised copy of each constant occurring in $\mathcal{D}$ (Line 2), and replicates in $\mathcal{D}_{\mathsf{an}}$ all atoms (unary and binary) of $\mathcal{D}$ on these new constants (Lines 3-4). Moreover, if $R(a, b)$ is in $\mathcal{D}$,

Algorithm 1. ComputeView

INPUT : A CQE-instance $\mathbf{I} = (\mathcal{O}, \mathcal{D}, \mathcal{P})$ with an RL$^-$ ontology $\mathcal{O}$
OUTPUT: A view $\mathcal{V}$ for $\mathbf{I}$

1 $\mathcal{D}_{\mathsf{an}} := \emptyset$;
2 **foreach** *constant* a *occurring in* $\mathcal{D}$ **do** create a fresh constant an_a;
3 **foreach** $A(a) \in \mathcal{D}$ **do** $\mathcal{D}_{\mathsf{an}} := \mathcal{D}_{\mathsf{an}} \cup \{A(an_a)\}$;
4 **foreach** $R(a,b) \in \mathcal{D}$ **do** $\mathcal{D}_{\mathsf{an}} := \mathcal{D}_{\mathsf{an}} \cup \{R(an_a, an_b), R(a, an_b),\ R(an_a, b)\}$;
5 $\mathcal{D}_{\mathsf{sat}} := \mathcal{D} \cup \mathcal{D}_{\mathsf{an}}$;
6 **foreach** *atom* β **s.t.** $\mathcal{O} \cup \mathcal{D} \cup \mathcal{D}_{\mathsf{an}} \models \beta$ **do** $\mathcal{D}_{\mathsf{sat}} := \mathcal{D}_{\mathsf{sat}} \cup \{\beta\}$;
7 $\mathcal{V} := \emptyset$;
8 **repeat**
9 Choose $\beta \in \mathcal{D}_{\mathsf{sat}}$ such that $\mathcal{O} \cup \mathcal{V} \cup \{\beta\} \not\models \alpha$ for each $\alpha \in \mathcal{P}$;
10 $\mathcal{V} := \mathcal{V} \cup \{\beta\}$
11 **until** *no such* $\beta \in \mathcal{D}_{\mathsf{sat}}$ *can be chosen*;
12 **return** $\mathcal{V}$.

then the algorithm also relates by R in $\mathcal{D}_{\mathsf{an}}$ the constant a to the anonymised copy an_b of b, and the anonymised copy an_a of a to b (Line 4). Then, the algorithm saturates the resulting dataset $\mathcal{D}_{\mathsf{sat}} = \mathcal{D} \cup \mathcal{D}_{\mathsf{an}}$ with all facts entailed by $\mathcal{O} \cup \mathcal{D}_{\mathsf{sat}}$ (Line 6). Finally, it enforces the policy $\mathcal{P}$ by computing a maximal subset $\mathcal{V}$ of the saturated dataset that respects $\mathcal{P}$ (Lines 8-11), which is finally returned as the output. Note, that different choices in Line 9 might lead to different outputs, i.e., algorithm ComputeView is non-deterministic. Indeed, for a given instance $\mathbf{I}$, there may be several view-based censors that are optimal, and each run of the algorithm ComputeView computes one of them; however, as we will see later on, any view leading to an optimal censor is computed by some run of the algorithm.

Example 25. When receiving our running example CQE-instance $\mathbf{I}_{ex}$ as input, the algorithm ComputeView computes the view given in Figure 2(b). In this case, the algorithm's output is independent from the choices made in Line 9, i.e., all possible runs of the algorithm lead to the same result.

To see how different runs of the algorithm might lead to different outputs, consider a CQE-instance $\mathbf{I} = (\mathcal{O}, \mathcal{D}, \mathcal{P})$ where $\mathcal{O} = \{A(x) \wedge B(x) \rightarrow C(x)\}$, $\mathcal{D} = \{A(a), B(a)\}$, and $\mathcal{P} = \{C(a)\}$. There are two possible runs of ComputeView on $\mathbf{I}$, which lead to two different views $\{A(a), A(an_a), B(an_a), C(an_a)\}$ and $\{B(a), A(an_a), B(an_a), C(an_a)\}$, respectively. ■

The following theorem establishes correctness and complexity of the algorithm.

Theorem 26. *Let* $\mathbf{I} = (\mathcal{O}, \mathcal{D}, \mathcal{P})$ *be a valid input of the algorithm* ComputeView.
(i) If $\mathcal{V}$ *is the result of a run of* ComputeView *on* $\mathbf{I}$ *then* $\mathsf{cens}_{\mathcal{V}}$ *is an optimal censor for* $\mathbf{I}$.

(ii) For each optimal censor cens *for* $\mathbf{I}$*, there exists a run of* ComputeView *on* $\mathbf{I}$ *that computes* $\mathcal{V}$ *such that* $\mathsf{cens} = \mathsf{cens}_{\mathcal{V}}$.

(iii) The algorithm can be implemented to run in time polynomial in $|\mathcal{D}| + |\mathcal{O}|$.

There is a couple of remarks about the proof of Theorem 26 that are worth to be made here. Polynomial complexity is a direct consequence of two facts: first, the size of $\mathcal{D}_{\mathsf{sat}}$ is polynomial in the size of $\mathcal{D}$ and $\mathcal{O}$; second, checking whether a ground atom can be entailed from an OWL 2 RL ontology and a dataset can be done in polynomial time in both the size of the ontology and dataset [7]. Optimality of $\mathsf{cens}_{\mathcal{V}}$ relies on the fact that $\mathcal{D}_{sat}$ captures all the possible matchings of an input query Q over the least Herbrand model of $\mathcal{O} \cup \mathcal{D}$; for this to be the case, the restrictions on rules imposed by RL$^-$ are the key.

7 Uniqueness of Optimal View-Based Censors

The fact that the output of ComputeView is not uniquely determined can be problematic. For example, a CQE system that relies on this algorithm needs to ensure that the same view is used in different user sessions, as well as for different users for which equivalent policies apply. One could, of course, determinise the output of the algorithm using application-dependent heuristics; however, there may not be a good reason for choosing one possible view over the others.

Our first proposal is to impose further restrictions to the ontology language. Example 25 suggests that existence of multiple views is related to the presence of conjunction in the bodies of rules, which suggests the restriction given next.

Definition 27 (Linear RL$^-$ ontology). *An* RL$^-$ *ontology* $\mathcal{O}$ *is* linear *if every rule in* $\mathcal{O}$ *contains exactly one atom in the body.*

Each RDFS ontology, such as $\mathcal{O}_{ex}$ in our running example, is also a linear RL$^-$ ontology; hence, linearity might not be too strict a restriction for many Semantic Web applications. Note also that linear RL$^-$ is a fragment of OWL 2 QL, in the sense that every linear RL$^-$ rule can be transformed into an equivalent OWL 2 QL axiom. In contrast, non-linear RL$^-$ is not captured by OWL 2 QL since it allows for conjunction in the body of rules. The following theorem shows that restricting ourselves to linear ontologies has the desired effect.

Theorem 28. *If* $\mathbf{I} = (\mathcal{O}, \mathcal{D}, \mathcal{P})$ *is a CQE-instance with* $\mathcal{O}$ *a linear* RL$^-$ *ontology; then, each run of* ComputeView *on* $\mathbf{I}$ *yields the same result.*

Corollary 29. *Let* $\mathbf{I} = (\mathcal{O}, \mathcal{D}, \mathcal{P})$ *be a CQE-instance with* $\mathcal{O}$ *a linear* RL$^-$ *ontology and let* $\mathcal{V}$ *be the result of a run of* ComputeView *on* $\mathbf{I}$*; then,* $\mathsf{cens}_{\mathcal{V}}$ *is the only optimal censor for* $\mathbf{I}$.

For applications where linearity is too strict, we can give up optimality in favour of uniqueness by taking the "intersection" of all optimal views.

Definition 30. *Let* $\mathbf{I} = (\mathcal{O}, \mathcal{D}, \mathcal{P})$ *be a CQE-instance with* $\mathcal{O}$ *an* RL^- *ontology. The* WIDTIO-view[1] *for* $\mathbf{I}$ *is the view defined as the following set of atoms:*

$$\{\textit{ground atom } \beta \mid \mathcal{O} \cup \mathcal{V} \models \beta \textit{ for each view } \mathcal{V} \textit{ for } \mathbf{I} \textit{ with } \mathsf{cens}_{\mathcal{V}} \textit{ optimal}\}.$$

The (non-optimal) censor based on the WIDTIO-view implements a "cautious" approach to confidentiality enforcement since it disregards all atoms that could possibly participate in the disclosure of an atom in $\mathcal{P}$. The following theorem shows, however, that this solution might be computationally expensive.

Theorem 31. *Deciding whether a ground atom* β *belongs to the WIDTIO-view for a CQE-instance* $\mathbf{I} = (\mathcal{O}, \mathcal{D}, \mathcal{P})$ *with* $\mathcal{O}$ *an* RL^- *ontology is* coNP-complete.

8 Related Work

The CQE paradigm was first proposed by Sichermann et al. [6], and was later extended by Biskup, Bonatti, Kraus and Subrahmanian [3,2,4,1]. CQE in the context of incomplete databases was studied by Biskup and Weibert [5]. These foundational works on CQE assume that both the information in the system and user queries are represented in propositional logic. Recently, Biskup and Bonatti studied CQE in relational databases where queries contain answer variables [12].

The formal study of data privacy and information hiding has received significant attention within the database community. Miklau and Suciu introduced *perfect privacy* in data exchange [13]. Rizvi et al. studied view-based authorisation mechanisms [14,15], and Deutsch et al. analysed the logical implications to privacy derived from publishing views of a database [16]. Formal data privacy frameworks have been proposed by Kifer et al. [17] and Evfimievski et. al. [18].

Privacy and information hiding in the context of ontologies has been investigated only recently. Information hiding at the schema level has been studied in [19,20]. Data privacy was studied for $\mathcal{EL}$ ontologies by Tao et al. [21]. Bao et al. introduced the notion of a privacy-preserving reasoner [22] and Stouppa et al. proposed a framework for data privacy in the context of $\mathcal{ALC}$ ontologies [23]. Finally, Calvanese et al. [24] proposed techniques for ontology access authorisation based on Zhang and Mendelzon's database authorisation views paradigm [15].

9 Conclusion and Future Work

We have proposed novel techniques for enforcing confidentiality in information systems that rely on RDF, SPARQL, and OWL 2 RL for representing data, queries, and ontologies, respectively. Our techniques ensure an optimal tradeoff between confidentiality and accessibility of information; furthermore, they can be efficiently implemented by relying on existing highly optimised OWL 2 RL triple stores. Our next step is to implement and test our algorithms, and we are

[1] This solution is related to the well-known "When In Doubt Through It Out" (WIDTIO) approach [11] to knowledge base update.

also planning to consider the problem of view maintenance for applications that require very frequent changes to the data, such as data streaming applications. Finally, we will study how our results could be extended to the case where the ontology is expressed in either the QL, or the EL profile of OWL 2.

References

1. Biskup, J., Bonatti, P.A.: Controlled Query Evaluation for Enforcing Confidentiality in Complete Information Systems. Int. J. Inf. Sec. 3(1), 14–27 (2004)
2. Biskup, J., Bonatti, P.A.: Lying Versus Refusal for Known Potential Secrets. Data Knowl. Eng. 38(2), 199–222 (2001)
3. Bonatti, P.A., Kraus, S., Subrahmanian, V.S.: Foundations of Secure Deductive Databases. IEEE Trans. Knowl. Data Eng. 7(3), 406–422 (1995)
4. Biskup, J.: For Unknown Secrecies Refusal Is Better than Lying. Data Knowl. Eng. 33(1), 1–23 (2000)
5. Biskup, J., Weibert, T.: Keeping Secrets in Incomplete Databases. Int. J. Inf. Sec. 7(3), 199–217 (2008)
6. Sicherman, G.L., de Jonge, W., van de Riet, R.P.: Answering Queries Without Revealing Secrets. ACM Trans. Database Syst. 8(1), 41–59 (1983)
7. Motik, B., Cuenca Grau, B., Horrocks, I., Wu, Z., Fokoue, A., Lutz, C.: OWL 2 Web Ontology Language Profiles, 2nd edn. W3C Recommendation (2012)
8. Bishop, B., Kiryakov, A., Ognyanoff, D., Peikov, I., Tashev, Z., Velkov, R.: OWLim: A Family of Scalable Semantic Repositories. Semantic Web J. 2(1), 33–42 (2011)
9. Wu, Z., Eadon, G., Das, S., Chong, E.I., Kolovski, V., Annamalai, M., Srinivasan, J.: Implementing an Inference Engine for RDFS/OWL Constructs and User-Defined Rules in Oracle. In: ICDE, pp. 1239–1248 (2008)
10. Motik, B., Patel-Schneider, P.F., Parsia, B.: OWL 2 Web Ontology Language Structural Specification and Functional-Style Syntax. W3C Recommendation (2012)
11. Eiter, T., Gottlob, G.: On the Complexity of Propositional Knowledge Base Revision, Updates, and Counterfactuals. In: PODS, pp. 261–273 (1992)
12. Biskup, J., Bonatti, P.: Controlled Query Evaluation with Open Queries for a Decidable Relational Submodel. Ann. Math. and Artif. Intell. 50(1-2), 39–77 (2007)
13. Miklau, G., Suciu, D.: A Formal Analysis of Information Disclosure in Data Exchange. J. Comput. Syst. Sci. 73(3), 507–534 (2007)
14. Rizvi, S., Mendelzon, A.O., Sudarshan, S., Roy, P.: Extending Query Rewriting Techniques for Fine-Grained Access Control. In: SIGMOD. ACM (2004)
15. Zhang, Z., Mendelzon, A.O.: Authorization Views and Conditional Query Containment. In: Eiter, T., Libkin, L. (eds.) ICDT 2005. LNCS, vol. 3363, pp. 259–273. Springer, Heidelberg (2005)
16. Deutsch, A., Papakonstantinou, Y.: Privacy in Database Publishing. In: Eiter, T., Libkin, L. (eds.) ICDT 2005. LNCS, vol. 3363, pp. 230–245. Springer, Heidelberg (2005)
17. Kifer, D., Machanavajjhala, A.: A Rigorous and Customizable Framework for Privacy. In: PODS, pp. 77–88 (2012)
18. Evfimievski, A.V., Fagin, R., Woodruff, D.P.: Epistemic Privacy. In: PODS, pp. 171–180. ACM (2008)
19. Konev, B., Walther, D., Wolter, F.: Forgetting and Uniform Interpolation in Large-Scale Description Logic Terminologies. In: IJCAI, pp. 830–835 (2009)

20. Cuenca Grau, B., Motik, B.: Reasoning over Ontologies with Hidden Content: The Import-by-Query Approach. J. Artif. Intell. Res. (JAIR) 45, 197–255 (2012)
21. Tao, J., Slutzki, G., Honavar, V.: Secrecy-Preserving Query Answering for Instance Checking in $\mathcal{EL}$. In: Hitzler, P., Lukasiewicz, T. (eds.) RR 2010. LNCS, vol. 6333, pp. 195–203. Springer, Heidelberg (2010)
22. Bao, J., Slutzki, G., Honavar, V.: Privacy-Preserving Reasoning on the Semantic Web. In: WI, pp. 791–797. IEEE Computer Society (2007)
23. Stouppa, P., Studer, T.: A Formal Model of Data Privacy. In: PSI (2007)
24. Calvanese, D., Giacomo, G.D., Lenzerini, M., Rosati, R.: View-based Query Answering over Description Logic Ontologies. In: KR. AAAI Press (2008)

Completeness Statements about RDF Data Sources and Their Use for Query Answering

Fariz Darari, Werner Nutt, Giuseppe Pirrò, and Simon Razniewski

Free University of Bozen-Bolzano, Bozen-Bolzano, Italy
fariz.darari@stud-inf.unibz.it, {nutt,pirro,razniewski}@inf.unibz.it

Abstract. With thousands of RDF data sources available on the Web covering disparate and possibly overlapping knowledge domains, the problem of providing high-level descriptions (in the form of metadata) of their content becomes crucial. In this paper we introduce a theoretical framework for describing data sources in terms of their completeness. We show how existing data sources can be described with completeness statements expressed in RDF. We then focus on the problem of the completeness of query answering over plain and RDFS data sources augmented with completeness statements. Finally, we present an extension of the completeness framework for federated data sources.

1 Introduction

The Resource Description Framework (RDF) [9] is the standard data model for the publishing and interlinking of data on the Web. It enables the making of *statements* about resources in the form of triples including a *subject*, a *predicate* and an *object*. Ontology languages such as RDF Schema (RDFS) and OWL provide the necessary underpinning for the creation of vocabularies to structure knowledge domains. RDF is now a reality; efforts like the Linked Open Data project [8] give a glimpse of the magnitude of RDF data today available online. The common path to access such huge amount of structured data is via SPARQL endpoints, that is, network locations that can be queried upon by using the SPARQL query language [5].

With thousands of RDF data sources covering possibly overlapping knowledge domains, the problem of providing high-level descriptions (in the form of metadata) of their content becomes crucial. Such descriptions will connect data publishers and consumers; publishers will advertise "what" is there inside a data source so that specialized applications can be created for data source discovering, cataloging, selection and so forth. Proposals like the `VoID` [1] vocabulary touched this aspect. With `VoID` it is possible to provide statistics about how many instances a particular *class* has, information about its SPARQL endpoint and links with other data sources, among the other things. However, `VoID` mainly focuses on providing *quantitative* information. We claim that toward comprehensive descriptions of data sources, *qualitative* information is crucial.

Related Work. Data quality is about the "fitness for use" of data and encompasses several dimensions such as accuracy, correctness and completeness.

H. Alani et al. (Eds.): ISWC 2013, Part I, LNCS 8218, pp. 66–83, 2013.

Fürber and Hepp [4] investigated data quality problems for RDF data originating from relational databases, while Wang et al. [19] focused on data cleansing. The problem of assessing completeness of Linked Data sources was discussed by Harth and Speiser [6]; here, completeness is defined in terms of *authoritativeness* of data sources, which is a purely syntactic property. Polleres et al. [16] defined a rule language where the need for completeness information emerges. Hartig et al. [7] discussed an approach to get more complete results of SPARQL queries over the Web of Linked Data. Their approach is based on traversing RDF links to discover relevant data during query execution. Still, the completeness of query answers cannot be guaranteed. In the relational databases world, completeness was first investigated by Motro [12] who provided a formalization of completeness of databases and queries. Halevy [11] studied the problem of how statements of completeness about a database related to query completeness. Recently, Razniewski and Nutt [17] provided a general solution to this problem, including a comprehensive study of the complexity of reasoning.

Indeed, the semantics of completeness is crucial also for RDF data sources distributed on the Web, where each data source is generally considered incomplete. To the best of our knowledge, the problem of formalizing the semantics of RDF data sources in terms of their completeness is open. Also from the more pragmatic point of view, there exist no comprehensive solutions enabling the characterization of data source in terms of completeness. As an example, with `VoID` it is not possible to express that, for instance, the data source IMDb is *complete for all movies directed by Tarantino*. Having the possibility to provide in a declarative and machine-readable way (in RDF), such kind of completeness statements paves the way toward a new generation of services for retrieving and consuming data. In this latter respect, the semantics of completeness statements interpreted by a reasoning engine can guarantee the completeness of query answering. We present a comprehensive application scenario in Section 2.

Contributions. This paper lays the foundation for the expression of completeness statements about RDF data sources. It can complement, with *qualitative* descriptions, existing proposals like `VoID` that mainly deal with *quantitative* descriptions. We develop a formalism and show its feasibility. The second goal of this paper is to show how completeness statements can be useful in practice. In this respect, we focus on the problem of query completeness. We believe that our research has both a theoretical and practical impact. On the theoretical side, we provide a formalization of completeness for RDF data sources and techniques to reason about the completeness of query answers in various settings, from plain RDF to federated data sources. From the practical side, completeness statements can be easily embedded in current descriptions of data sources and thus readily used. Finally, we want to point out that our completeness framework has been implemented in the `CoRNER` system, which is available for download[1].

Outline. In Section 2 we discuss a real world scenario and provide a high level overview of the completeness framework. Section 3 after providing some

[1] `http://rdfcorner.wordpress.com/`

background introduces a formalization of the completeness problem for RDF data sources. This section also describes how completeness statements can be represented in RDF. In Section 4 we discuss how completeness statements can be used in query answering when considering a single data source at a time. In Section 5 we challenge query completeness in federated data sources. Section 6 contains a discussion and Section 7 the conclusions.

2 Motivating Scenario

In this section we motivate the need of formalizing and expressing completeness statements in a machine-readable way. Moreover we show how completeness statement are useful for query answering. We start our discussion with a real data source available on the Web. Fig. 1 shows a screenshot taken from the IMDb website. The page is about the movie Reservoir Dogs; in particular it lists the cast and crew of the movie. For instance, it says that Tarantino was not only the director and writer of the movie but also the character Mr. Brown. As it can be noted, the data source includes a "completeness statement", which says that the page is *complete for all cast and crew members of the movie*. The availability of such statement increases the potential value of the data source. In particular, users who were looking for information about the cast of this movie and found this page can prefer it to other pages since, assuming the truth of the statement, all they need is here.

Fig. 1. A *completeness statement* in IMDb as of 7 May 2013. It says that the source is complete for the cast and crew of the movie *Reservoir Dogs*.

The problem with such kind of statements, expressed in natural language, is that they cannot be automatically processed, thus hindering their applicability, for instance, in query answering. Indeed, the interpretation of the statement "verified as complete" is left to the user. On the other hand, a reasoning and querying engine when requested to provide information about the cast and crew

members of Reservoir Dogs could have leveraged such statement and inform the user about the completeness of the results.

Other examples of Web data sources that already provide completeness statements are OpenStreetMap[2] and Wikipedia, which has, for instance, a complete list of works attributed to *Vermeer* and works by *Shakespeare* or a complete list of Olympic medalists in archery from 1900 to 2012. If such statements were exploited by machines, one would expect that there would be an incentive to publish them.

Machine-Readable Statements. In the RDF and Linked Data context with generally incomplete and possibly overlapping data sources and where *"anyone can say anything about any topic and publish it anywhere"* [9] having the possibility to express completeness statements becomes an essential aspect. The machine-readable nature of RDF enables to deal with the problems discussed in the example about IMDb; completeness statements can be represented in RDF. As an example, the high-level description of a data source like DBpedia could include, for instance, the fact that it is complete for all of Quentin Tarantino's movies. Fig. 2 shows how the data source DBpedia can be complemented with completeness statements expressed in our formalism. Here we give a high level presentation of the completeness framework; details on the theoretical framework supporting it are given in Section 3.

```
@prefix       c: <http://inf.unibz.it/ontologies/completeness#> .
@prefix     rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix    rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix  schema: <http://schema.org/> .
@prefix    spin: <http://spinrdf.org/sp#> .
@prefix     dbp: <http://dbpedia.org/resource/> .
@prefix    void: <http://rdfs.org/ns/void#> .
@prefix      dv: <http://dbpedia.org/void/> .

dv:dbpdataset rdf:type void:Dataset .

dv:dbpdataset rdfs:comment "This document provides completeness statements
about the dbpedia.org datasource" .

dv:dbpdataset c:hasComplStmt dv:st1.
dv:st1        c:hasPattern    [c:subject   [spin:varName "m"];
                                           c:predicate rdf:type;
                                           c:object    schema:Movie     ].
dv:st1        c:hasPattern    [c:subject   [spin:varName "m"];
                                           c:predicate schema:director;
                                           c:object    dbp:Tarantino].
dv:st1        rdfs:comment "This completeness statement indicates that
dbpedia.org is complete for all movies directed by Tarantino".
```

Fig. 2. An example of completeness statement about `dbpedia.org`

A simple statement can be thought of as a SPARQL Basic Graph Pattern (BGP). The BGP `(?m rdf:type schema:Movie).(?m schema:director dbp:Tarantino)`, for instance, expresses the fact that `dbpedia.org` is complete for all movies directed by Tarantino. In the figure, this information is

[2] `http://wiki.openstreetmap.org/wiki/Hall_of_Fame/Streets_complete`

represented by using an ad-hoc completeness vocabulary (see Section 3.2) with some properties taken from the SPIN[3] vocabulary.

Query Completeness. The availability of completeness statements about data sources is useful in different tasks, including data integration, data source discovery and query answering. In this paper we will focus on how to leverage completeness statements for query answering. The research question we address is how to assess whether available data sources with different degree of completeness can ensure the completeness of query answers. Consider the scenario depicted in Fig. 3 where the data sources DBpedia and LinkedMDB are described in terms of their completeness. The Web user Syd wants to pose the query Q to the SPARQL endpoints of these two data sources asking for *all movies directed by Tarantino in which Tarantino also starred*. By leveraging the completeness statements, the query engines at the two endpoints could tell Syd whether the answer to his query is complete or not. For instance, although DBpedia is complete for all of Tarantino's movies (see Fig. 2) nothing can be said about his participation as an actor in these movies (which is required in the query). Indeed, at the time of writing this paper, DBpedia is actually incomplete; this is because in the description of the movie Reservoir Dogs the fact is missing that Tarantino was the character Mr. Brown (and from Fig. 1 we know that this is the case). On the other hand, LinkedMDB, the RDF counterpart of IMDb, can provide a complete answer. Indeed, with our framework it is possible to express in RDF the completeness statement available in natural language in Fig. 1. This statement has then been used by the `CoRNER` reasoning engine, implementing our formal framework, to state the completeness of the query.

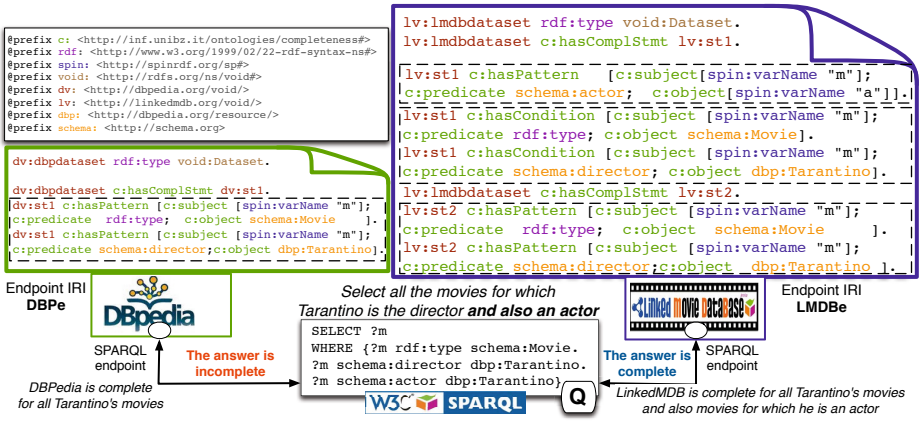


Fig. 3. Completeness statements and their usage for query answering

In this specific case, LinkedMDB can guarantee the completeness of the query answer because it contains all the actors in Tarantino's movies (represented by

[3] `http://spinrdf.org/sp.html#sp-variables`

the statement `lv:st1`) in addition to the Tarantino's movies themselves (represented by the statement `lv:st2`). Note that the statement `lv:st1` includes two parts: (i) the pattern, which is expressed via the BGP `(?m, schema:actor, ?a)` and (ii) the conditions, that is, the BGP `(?m, rdf:type, schema:Movie).(?m, schema:director, dbp:Tarantino)`. Indeed, a completeness statement allows one to say that a certain part (i.e., with respect to some conditions) of data is complete, or in other words, it can be used to state that a data source contains all triples in a pattern P_1 that satisfy a condition P_2. The detailed explanation and the semantics of completeness statements can be found in Section 3.

Application Scenarios. Completeness statements are particularly useful for data collections such as works of an artist, cities in countries, election results, census data and so forth. Completeness statements have wide applicability. *Source selection*: as an example for address verification, one needs a complete set of street names; for Hamburg, Dresden, and other cities in Germany, OpenStreetMap can be used because completeness is asserted. *Search Optimization*: a user wants to look for movies by Tarantino in 2008. By having completeness statements in IMDb about these movies, a search engine could stop after finding this specific source without the need to consult other sources.

3 Formal Framework

In the following, we remind the reader of RDF and SPARQL, formalize our framework and show how completeness information can be expressed in RDF.

RDF and SPARQL. We assume that there are three pairwise disjoint infinite sets I (*IRIs*), L (*literals*) and V (*variables*). We collectively refer to IRIs and literals as *RDF terms* or simply *terms*. A tuple $(s, p, o) \in I \times I \times (I \cup L)$ is called an *RDF triple* (or a *triple*), where s is the *subject*, p the *predicate* and o the *object* of the triple. An *RDF graph* or *data source* consists of a finite set of triples [9]. For simplicity, we omit namespaces for the abstract representation of RDF graphs.

The standard query language for RDF is SPARQL. The basic building blocks of a SPARQL query are *triple patterns,* which resemble RDF triples, except that in each position also variables are allowed. SPARQL queries include *basic graph patterns* (BGP), built using the `AND` operator, and more sophisticated operators, including `OPT`, `FILTER`, `UNION` and so forth. In this paper we consider the operators `AND` and `OPT`. Moreover, we also consider the result modifier `DISTINCT`. Evaluating a graph pattern P over an RDF graph G results in a set of mappings from the variables in P to terms, denoted as $[\![P]\!]_G$. Further information about SPARQL can be found in [14].

SPARQL queries come as `SELECT`, `ASK`, or `CONSTRUCT` queries. A `SELECT` query has the abstract form (W, P), where P is a graph pattern and W is a subset of the variables in P. A `SELECT` query $Q = (W, P)$ is evaluated over a graph G by restricting the mappings in $[\![P]\!]_G$ to the variables in W. The result is denoted as $[\![Q]\!]_G$. Syntactically, an `ASK` query is a special case of a `SELECT` query where

W is empty. For an `ASK` query Q, we write also $[\![Q]\!]_G = \texttt{true}$ if $[\![Q]\!]_G \neq \emptyset$, and $[\![Q]\!]_G = \texttt{false}$ otherwise. A `CONSTRUCT` query has the abstract form (P_1, P_2), where P_1 is a BGP and P_2 is a graph pattern. In this paper, we only use `CONSTRUCT` queries where also P_2 is a BGP. The result of evaluating $Q = (P_1, P_2)$ over G is the graph $[\![Q]\!]_G$, that is obtained by instantiating the pattern P_1 with all the mappings in $[\![P_2]\!]_G$.

Later on, we will distinguish between three classes of queries: (i) Basic queries, that is, queries (W, P) where P is a BGP and which return bags of mappings (as it is the default in SPARQL), (ii) `DISTINCT` queries, that is, queries $(W, P)^d$ where P is a BGP and which return sets of mappings, and (iii) `OPT` queries, that is, queries (W, P) without projection ($W = var(P)$) where P is a graph pattern with `OPT`.

3.1 Completeness Statements and Query Completeness

We are interested in formalizing when a query is complete over a potentially incomplete data source and in describing which parts of such a source are complete. When talking about the completeness of a source, one implicitly compares the information *available* in the source with what holds in the world and therefore should *ideally* be also present in the source. In this paper, we only consider sources that may miss information, but do not contain wrong information.

Definition 1 *(**Incomplete Data Source**). We identify data sources with RDF graphs. Then, adapting a notion introduced by Motro in [12], we define an incomplete data source as a pair $\mathcal{G} = (G^a, G^i)$ of two graphs, where $G^a \subseteq G^i$. We call G^a the* available *graph and G^i the* ideal *graph.*

Example 2 *(**Incomplete Data Source**).* Consider the DBpedia data source and suppose that the only movies directed by Tarantino are Reservoir Dogs, Pulp Fiction, and Kill Bill, and that Tarantino was starred exactly in the movies Desperado, Reservoir Dogs, and Pulp Fiction. For the sake of example, suppose also the fact that he was starred in Reservoir Dogs is missing in DBpedia[4]. Using Definition 1, we can formalize the incompleteness of the DBpedia data source $\mathcal{G}_{dbp}$ as:

$$
\begin{aligned}
G^a_{dbp} = \{&(\mathit{reservoirDogs}, \mathit{director}, \mathit{tarantino}), (\mathit{pulpFiction}, \mathit{director}, \mathit{tarantino}),\\
&(\mathit{killBill}, \mathit{director}, \mathit{tarantino}), (\mathit{desperado}, \mathit{actor}, \mathit{tarantino}),\\
&(\mathit{pulpFiction}, \mathit{actor}, \mathit{tarantino}), (\mathit{desperado}, \mathit{type}, \mathit{Movie}),\\
&(\mathit{reservoirDogs}, \mathit{type}, \mathit{Movie}), (\mathit{pulpFiction}, \mathit{type}, \mathit{Movie}), (\mathit{killBill}, \mathit{type}, \mathit{Movie})\}\\
G^i_{dbp} = G^a_{dbp} \cup \{&(\mathit{reservoirDogs}, \mathit{actor}, \mathit{tarantino})\}
\end{aligned}
$$

We now introduce *completeness statements*, which are used to denote the partial completeness of a data source, that is, they describe for which parts the ideal and available graph coincide.

[4] As it was the case on 7 May 2013.

Definition 3 ***(Completeness Statement).*** *A completeness statement* $Compl(P_1 \mid P_2)$ *includes:* P_1 *a non-empty BGP and* P_2 *a BGP. We call* P_1 *the* pattern *and* P_2 *the* condition *of the completeness statement.*

For example, we express that a source is complete for all pairs of triples that say "?m *is a movie and* ?m *is directed by Tarantino*" using the statement

$$C_{dir} = Compl((?m, type, Movie), (?m, director, tarantino) \mid \emptyset), \tag{1}$$

whose pattern matches all such pairs and whose condition is empty. To express that a source is complete for all triples about actors in movies directed by Tarantino, we use

$$C_{act} = Compl((?m, actor, ?a) \mid (?m, director, tarantino), (?m, type, Movie)), \tag{2}$$

whose pattern matches triples about actors and the condition restricts the actors to movies directed by Tarantino. The condition in C_{act} means that the data source does not necessarily contain triples of the form $(?m, director, tarantino)$ and $(?m, type, Movie)$. Moving the condition to the pattern imposes that the data source contains the triples.

We now define when a completeness statement is satisfied by an incomplete data source. To a statement $C = Compl(P_1 \mid P_2)$, we associate the `CONSTRUCT` query $Q_C = (P_1, P_1 \cup P_2)$. Note that, given a graph G, the query Q_C returns those instantiations of the pattern P_1 that are present in G together with an instantiation of the condition. For example, the query $Q_{C_{act}}$ returns all the acting information of Tarantino movies in G.

Definition 4 ***(Satisfaction of Completeness Statements).*** *For an incomplete data source* $\mathcal{G} = (G^a, G^i)$*, the statement* C *is satisfied by* $\mathcal{G}$*, written* $\mathcal{G} \models C$*, if* $[\![Q_C]\!]_{G^i} \subseteq G^a$ *holds.*

To see that the statement C_{dir} is satisfied by $\mathcal{G}_{dbp}$, observe that the query $Q_{C_{dir}}$ returns over G^i_{dbp} all triples with the predicate *actor* and all *type* triples for Tarantino movies, and that all these triples are also in G^a_{dbp}. However, C_{act} is *not* satisfied by $\mathcal{G}_{dbp}$, because $Q_{C_{act}}$ returns over G^i_{dbp} the triple $(reservoirDogs, actor, tarantino)$, which is not in G^a_{dbp}.

When querying a potentially incomplete data source, we would like to know whether at least the answer to our query is complete. For instance, when querying DBpedia for movies starring Tarantino, it would be interesting to know whether we really get all such movies, that is, whether our query is complete over DBpedia. We next formalize query completeness with respect to incomplete data sources.

Definition 5 ***(Query Completeness).*** *Let* Q *be a* `SELECT` *query. To express that* Q is complete*, we write* $Compl(Q)$*. An incomplete data source* $\mathcal{G} = (G^a, G^i)$ *satisfies the expression* $Compl(Q)$*, if* Q *returns the same result over* G^a *as it does over* G^i*, that is* $[\![Q]\!]_{G^a} = [\![Q]\!]_{G^i}$*. In this case we write* $\mathcal{G} \models Compl(Q)$*.*

Example 6 ***(Query Completeness).*** Consider the incomplete data source $\mathcal{G}_{dbp}$ and the two queries Q_{dir}, asking for all movies directed by Tarantino, and $Q_{dir+act}$, asking for all movies, both directed by and starring Tarantino:

$$Q_{dir} = (\{\, ?m \,\}, \{\, (?m, type, Movie), (?m, director, tarantino) \,\})$$
$$Q_{dir+act} = (\{\, ?m \,\}, \{\, (?m, type, Movie), (?m, director, tarantino), (?m, actor, tarantino) \,\}).$$

Then, it holds that Q_{dir} is complete over $\mathcal{G}_{dbp}$ while $Q_{dir+act}$ is not. Later on, we show how to deduce query completeness from completeness statements.

3.2 RDF Representation of Completeness Statements

Practically, completeness statements should be compliant with the existing ways of giving metadata about data sources, for instance, by enriching the `VoID` description [1]. Therefore, it is essential to express completeness statements in RDF itself. Suppose we want to express that LinkedMDB satisfies the statement:

$$C_{act} = Compl((?m, actor, ?a) \mid (?m, type, Movie), (?m, director, tarantino)).$$

Then, we need a vocabulary to say that this is a statement about LinkedMDB, which triple patterns make up its pattern, and which its condition. We also need the vocabulary to represent the constituents of the triple patterns, namely subject, predicate, and object of a pattern. Therefore, we introduce the property names whose meaning is intuitive:

`hasComplStmt`, `hasPattern`, `hasCondition`, `subject`, `predicate`, `object`

If the constituent of a triple pattern is a term (an IRI or a literal), then it can be specified directly in RDF. Since this is not possible for variables, we represent a variable by a resource that has a literal value for the property `varName`. Now, we can represent C_{act} in RDF as the resource `lv:st1` described in Figure 3.

More generally, consider a completeness statement $Compl(P_1 \mid P_2)$, where $P_1 = \{\, t_1, \ldots, t_n \,\}$ and $P_2 = \{\, t_{n+1}, \ldots, t_m \,\}$ and each t_i, $1 \leq i \leq m$, is a triple pattern. Then the statement is represented using a resource for the statement and a resource for each of the t_i that is linked to the statement resource by the property `hasPattern` or `hasCondition`, respectively. The constituents of each t_i are linked to t_i's resource in the same way via `subject`, `predicate`, and `object`. All resources can be either IRIs or blank nodes.

4 Completeness Reasoning over a Single Data Source

In this section, we show how completeness statements can be used to judge whether a query will return a complete answer or not. We first focus on completeness statements that hold on a *single* data source, while completeness statements in the federated setting are discussed in Section 5.

Problem Definition. Let $\mathbf{C}$ be a set of completeness statements and Q be a `SELECT` query. We say that $\mathbf{C}$ *entails the completeness of* Q, written $\mathbf{C} \models \mathit{Compl}(Q)$, if any incomplete data source that satisfies $\mathbf{C}$ also satisfies $\mathit{Compl}(Q)$.

Example 7. Consider C_{dir} from (1). Whenever an incomplete data source $\mathcal{G}$ satisfies C_{dir}, then G^a contains all triples about movies directed by Tarantino, which is exactly the information needed to answer query Q_{dir} from Example 6. Thus, $\{ C_{dir} \} \models \mathit{Compl}(Q_{dir})$. This may not be enough to completely answer $Q_{dir+act}$, thus $\{ C_{dir} \} \not\models \mathit{Compl}(Q_{dir+act})$. We will now see how this intuitive reasoning can be formalized.

4.1 Completeness Entailment for Basic Queries

To characterize completeness entailment, we use the fact that completeness statements have a correspondence in `CONSTRUCT` queries. For any set $\mathbf{C}$ of completeness statements we define the operator $T_{\mathbf{C}}$ that maps graphs to graphs:

$$T_{\mathbf{C}}(G) = \bigcup_{C \in \mathbf{C}} Q_C(G)$$

Notice that for any graph G, the pair $(T_{\mathbf{C}}(G), G)$ is an incomplete data source satisfying $\mathbf{C}$ and $T_{\mathbf{C}}(G)$ is the smallest set (wrt. set inclusion) for which this holds.

Example 8 ***(Completeness Entailment).*** Consider the set of completeness statements $\mathbf{C}_{dir,act} = \{ C_{dir}, C_{act} \}$ and the query $Q_{dir+act}$. Recall that the query has the form $Q_{dir+act} = (\{ ?m \}, P_{dir+act})$, where $P_{dir+act} = \{ (?m, \mathit{type}, \mathit{Movie}), (?m, \mathit{director}, \mathit{tarantino}), (?m, \mathit{actor}, \mathit{tarantino}) \}$. We want to check whether these statements entail the completeness of $Q_{dir+act}$, that is, whether $\mathbf{C}_{dir,act} \models \mathit{Compl}(Q_{dir+act})$ holds. Suppose that $\mathcal{G} = (G^a, G^i)$ satisfies $\mathbf{C}_{dir,act}$. Suppose also that $Q_{dir+act}$ returns a mapping $\mu = \{ ?m \mapsto m' \}$ over G^i for some term m'. Then G^i contains $\mu P_{dir+act}$, the instantiation by μ of the BGP of our query, consisting of the three triples $(m', \mathit{type}, \mathit{Movie})$, $(m', \mathit{director}, \mathit{tarantino})$, and $(m', \mathit{actor}, \mathit{tarantino})$.

The `CONSTRUCT` query $Q_{C_{dir}}$, corresponding to our first completeness statement, returns over $\mu P_{dir+act}$ the two triples $(m', \mathit{type}, \mathit{Movie})$ and $(m', \mathit{director}, \mathit{tarantino})$, while the `CONSTRUCT` query $Q_{C_{act}}$, corresponding to the second completeness statement, returns the triple $(m', \mathit{actor}, \mathit{tarantino})$. Thus, all triples in $\mu P_{dir+act}$ have been reconstructed by $T_{\mathbf{C}_{dir,act}}$ from $\mu P_{dir+act}$.

Now, we have $\mu P_{dir+act} = T_{\mathbf{C}_{dir,act}}(\mu P_{dir+act}) \subseteq T_{\mathbf{C}_{dir,act}}(G^i) \subseteq G^a$, where the last inclusion holds due to $\mathcal{G} \models \mathbf{C}_{dir,act}$. Therefore, our query $Q_{dir+act}$ returns the mapping μ also over G^a. Since μ and $\mathcal{G}$ were arbitrary, this shows that $\mathbf{C}_{dir,act} \models \mathit{Compl}(Q_{dir+act})$ holds.

In summary, in Example 8 we have reasoned about a set of completeness statements $\mathbf{C}$ and a query $Q = (W, P)$. We have considered a generic mapping μ, defined on the variables of P, and applied it to P, thus obtaining a graph μP.

Then we have verified that $\mu P = T_{\mathbf{C}}(\mu P)$. From this, we could conclude that for every incomplete data source $\mathcal{G} = (G^a, G^i)$ we have that $[\![Q]\!]_{G^a} = [\![Q]\!]_{G^i}$. Next, we make this approach formal.

Definition 9 ***(Prototypical Graph).*** *Let (W, P) be a query. The* freeze mapping *$\tilde{id}$ is defined as mapping each variable v in P to a new IRI $\tilde{v}$. Instantiating the graph pattern P with $\tilde{id}$ yields the RDF graph $\tilde{P} := \tilde{id}\, P$, which we call the prototypical graph of P.*

Now we can generalize the reasoning from above to a generic completeness check.

Theorem 10 (Completeness of Basic Queries). *Let $\boldsymbol{C}$ be a set of completeness statements and let $Q = (W, P)$ be a basic query. Then*

$$\boldsymbol{C} \models \mathit{Compl}(Q) \qquad \text{if and only if} \qquad \tilde{P} = T_{\boldsymbol{C}}(\tilde{P}).$$

Proof. (Sketch) "⇒" If $\tilde{P} \neq T_{\mathbf{C}}(\tilde{P})$, then the pair $(T_{\mathbf{C}}(\tilde{P}), \tilde{P})$ is a counterexample for the entailment. It satisfies $\mathbf{C}$, but does not satisfy $\mathit{Compl}(Q)$ because the freeze mapping $\tilde{id}$ cannot be retrieved by P over the available graph $T_{\mathbf{C}}(\tilde{P})$.

"⇐" If all triples of the pattern $\tilde{P}$ are preserved by $T_{\mathbf{C}}$, then this serves as a proof that in any incomplete data source all triples that are used to compute a mapping in the ideal graph are also present in the available graph.

Queries with `DISTINCT`. Basic queries return bags of answers (i.e., they may contain duplicates), while `DISTINCT` eliminates duplicates. For a query Q involving `DISTINCT`, the difference to the characterization in Theorem 10 is that instead of retrieving the full pattern $\tilde{P}$ after applying $T_{\mathbf{C}}$, we only check whether sufficient parts of $\tilde{P}$ are preserved that still allow to retrieve the freeze mapping on the distinguished variables of Q.

4.2 Completeness of Queries with the `OPT` Operator

One interesting feature of SPARQL is the `OPT` ("optional") operator. With `OPT` one can specify that parts of a query are only evaluated if an evaluation is possible, similarly to an outer join in SQL. For example, when querying for movies, one can also ask for the prizes they won, if any. The `OPT` operator is used substantially in practice [15]. Intuitively, the mappings for a pattern (P_1 `OPT` P_2) are computed as the union of all the bindings of P_1 together with the bindings for P_2 that are valid extensions, and including those bindings of P_1 that have no binding for P_2 that is a valid extension. For a formal definition of the semantics of queries with the `OPT` operator, see [10]. Completeness entailment for queries with `OPT` differs from that of queries without.

Example 11 ***(Completeness with `OPT`).*** Consider the following query with `OPT` $Q_{maw} = ((?m, \mathit{type}, \mathit{Movie})$ `OPT` $(?m, \mathit{award}, ?aw))$, asking for all movies and if available, also their awards. Consider also $C_{aw} = \mathit{Compl}((?m, \mathit{type}, \mathit{Movie}), (?m, \mathit{award}, ?aw) \mid \emptyset)$, the completeness statement

that expresses that all movies that have an award are complete and all awards of movies are complete. If the query Q_{maw} used `AND` instead of `OPT`, then its completeness could be entailed by C_{aw}. However with `OPT` in Q_{maw}, more completeness is required: Also those movies have to be complete that do not have an award. Thus, C_{aw} alone does not entail the completeness of Q_{maw}.

If one uses `OPT` without restrictions, unintuitive queries may result. Pérez et al. have introduced the class of so-called *well-designed graph patterns* that avoid anomalies that may otherwise occur [14]. Formally, a graph pattern P is well-designed if for every subpattern $P' = (P_1 \texttt{ OPT } P_2)$ of P and for every variable $?X$ occurring in P, the following condition holds: if $?X$ occurs both inside P_2 and outside P', then it also occurs in P_1. We restrict ourselves in the following to `OPT` queries with well-designed patterns, which we call well-designed queries. Graph patterns with `OPT` have a hierarchical structure that can be made explicit by so-called pattern trees. A *pattern tree* $\mathcal{T}$ is a pair $(T, \mathcal{P})$, where (i) $T = (N, E, r)$ is a tree with node set N, edge set E, and root $r \in N$, and (ii) $\mathcal{P}$ is a labeling function that associates to each node $n \in N$ a BGP $\mathcal{P}(n)$. We construct for each pattern P a corresponding pattern tree $\mathcal{T}$. Any `OPT`-pattern can be translated into a pattern tree and vice versa [10]. As an example, consider a pattern $((P_1 \texttt{ OPT } P_2) \texttt{ OPT } (P_3 \texttt{ OPT } P_4))$, where P_1 to P_4 are BGPs. Its corresponding pattern tree would have a root node labeled with P_1, two child nodes labeled with P_2 and P_3, respectively, and the P_3 node would have another child labeled with P_4.

Patterns and pattern trees can contain redundant triples. Letelier et al. [10] have shown that for every pattern tree $\mathcal{T}$ one can construct in polynomial time an equivalent well-designed pattern tree $\mathcal{T}^{NR}$ without redundant triples, which is called the NR-normal form of $\mathcal{T}$. For every node n in $\mathcal{T}$ we define the branch pattern P_n of n as the union of the labels of all nodes on the path from n to the root of $\mathcal{T}$. Then the *branch query* Q_n of n has the form (W_n, P_n), where $W_n = var(P_n)$.

Theorem 12 (Completeness of `OPT`-Queries). *Let $\boldsymbol{C}$ be a set of completeness statements. Let $Q = (W, P)$ be a well-designed `OPT`-query and $\mathcal{T}$ be an equivalent pattern tree in NR-normal form. Then*

$$\boldsymbol{C} \models Compl(Q) \quad \textit{iff} \quad \boldsymbol{C} \models Compl(Q_n) \quad \textit{for all branch queries } Q_n \textit{ of } \mathcal{T}.$$

Technically, this theorem allows to reduce completeness checking for an `OPT` query to linearly many completeness checks for basic queries.

4.3 Completeness Entailment under RDFS Semantics

RDFS (RDF Schema) is a simple ontology language that is widely used for RDF data [3]. RDFS information can allow additional inference about data and needs to be taken into account during completeness entailment.

Example 13 ***(RDF vs. RDFS).*** Consider the query $Q_{film} = (\{ ?m \}, \{ (?m, type, film) \})$, asking for all films, and the completeness statement

$C_{movie} = Compl((?m, type, movie) \mid \emptyset)$ saying that we are complete for all movies. A priori, we cannot conclude that C_{movie} entails the completeness of Q_{film}, because we do not know about the relationship between films and movies. When considering the RDFS statements $(film, subclass, movie)$ and $(movie, subclass, film)$ saying that all movies and films are equivalent, we can conclude that $\{C_{movie}\} \models Compl(Q_{film})$.

In the following, we rely on ρDF, which formalizes the core of RDFS [13]. The ρDF vocabulary contains the terms *subproperty*, *subclass*, *domain*, *range* and *type*. A *schema graph* S is a set of triples built using any of the ρDF terms, except *type*, as predicates.

We assume that schema information is not lost in incomplete data sources. Hence, for incomplete data sources it is possible to extract their ρDF schema into a separate graph. The *closure of a graph* G, that is, $cl_S(G)$ wrt. a schema S is the set of all triples that are entailed. The computation of this closure can be reduced to the computation of the closure of a single graph that contains both schema and non-schema triples as $cl_S(G) = cl(S \cup G)$. We now say that a set $\mathbf{C}$ of completeness statements *entails* the completeness of a query Q wrt. a ρDF schema graph S, if for all incomplete data sources (G^a, G^i) it holds that if $(cl_S(G^a), cl_S(G^i))$ satisfies $\mathbf{C}$ then it also satisfies $Compl(Q)$.

Therefore, the main difference to the previous entailment procedures is that the closure is computed to obtain entailed triples before and after the completeness operator $T_{\mathbf{C}}$ is applied. For a set of completeness statements $\mathbf{C}$ and a schema graph S, let $T^S_{\mathbf{C}}$ denote the function composition $cl_S \circ T_{\mathbf{C}} \circ cl_S$. Then the following holds.

Theorem 14 (Completeness under RDFS). *Let $\boldsymbol{C}$ be a set of completeness statements, $Q = (W, P)$ a basic query, and S a schema graph. Then*

$$\boldsymbol{C} \models_S Compl(Q) \qquad \text{if and only if} \qquad \tilde{P} \subseteq T^S_{\mathbf{C}}(\tilde{P}).$$

5 Completeness Reasoning over Federated Data Sources

Data on the Web is intrinsically distributed. Hence, the single-source query mechanism provided by SPARQL has been extended to deal with multiple data sources. In particular, the recent SPARQL 1.1 specification introduces the notion of query *federation* [18]. A federated query is a SPARQL query that is evaluated across several data sources, the SPARQL endpoints of which can be specified in the query.

So far, we have studied the problem of querying a *single* data source augmented with completeness statements. The federated scenario calls for an extension of the completeness framework discussed in Section 4. Indeed, the completeness statements available about each data source involved in the evaluation of a federated query must be considered to check the completeness of the federated query. This section discusses this aspect and presents an approach to

check whether the completeness of a non-federated query (i.e., a query without `SERVICE` operators) can be ensured with respect to the completeness statements on each data source. We also study the problem of rewriting a non-federated query into a federated version in the case in which the query is complete.

Federated SPARQL Queries. Before introducing the extension of the completeness framework, we formalize the notion of federated SPARQL queries. A federated query is a SPARQL query executed over a *federated graph*. Formally speaking, a federated graph is a family of RDF graphs $\bar{G} = (G_j)_{j \in J}$ where J is a set of IRIs. A federated SPARQL query (as for the case of a non-federated query) can be a `SELECT` or an `ASK` query [2]. In what follows, we focus on the conjunctive fragment (i.e., the `AND` fragment) of SPARQL with the inclusion of the `SERVICE` operator. Non-federated SPARQL queries are evaluated over graphs. In the federated scenario, queries are evaluated over a pair $(i, \bar{G})$, where the first component is an IRI associated to the initial SPARQL endpoint, and the second component is a federated graph. The semantics of graph patterns with `AND` and `SERVICE` operators is defined as follows:

$$\begin{aligned} [\![t]\!]_{(i,\bar{G})} &= [\![t]\!]_{G_i} \\ [\![P_1 \texttt{ AND } P_2]\!]_{(i,\bar{G})} &= [\![P_1]\!]_{(i,\bar{G})} \bowtie [\![P_2]\!]_{(i,\bar{G})} \\ [\![(\texttt{SERVICE } j \ P)]\!]_{(i,\bar{G})} &= [\![P]\!]_{(j,\bar{G})} \end{aligned}$$

where t ranges over all triple patterns and P, P_1, P_2 range over all graph patterns with `AND` and `SERVICE` operators. We denote federated queries as $\bar{Q}$.

5.1 Federated Completeness Reasoning Framework

We now extend our completeness reasoning framework to the federated setting. We assume from now on that the set of IRIs J is fixed and all indices are drawn from J.

Definition 15 (Incomplete Federated Data Source). *An* incomplete federated data source *(or incomplete FDS, for short) is a pair $\bar{\mathcal{G}} = (\bar{G}^a, G^i)$, consisting of an* available federated graph $\bar{G}^a = (G^a_j)_{j \in J}$ *and an* ideal graph G^i*, such that $G^a_j \subseteq G^i$ for all $j \in J$.*

This captures the intuition that the ideal graph represents all the facts that hold in the world, while each source contains a part of those facts. Note that the graphs of the sources may overlap, as is the case on the Web. Next, we adapt completeness statements so that they talk about a specific source.

Definition 16 (Indexed Completeness Statements). *An* indexed completeness statement *is a pair (C, k) where C is a completeness statement and $k \in J$ is an IRI. An indexed completeness statement is satisfied by an incomplete FDS if it is satisfied by the incomplete data source corresponding to the index, that is,*

$$((G^a_j)_{j \in J}, G^i) \models_{fed} (C, k) \quad \textit{iff} \quad (G^a_k, G^i) \models C.$$

This definition is naturally extended to sets $\bar{\boldsymbol{C}}$ of indexed completeness statements.

We associate to each federated query, federated graph, incomplete FDSs, and set of indexed completeness statements a non-federated version, the flattening.

Definition 17 (Flattening). *The* flattening $\bar{Q}^{fl}$ *of a federated query* $\bar{Q}$ *is obtained from* $\bar{Q}$ *by replacing recursively each occurrence of a service call* (`SERVICE` j P) *with the pattern* P. *The* flattening $\bar{G}^{fl}$ *of a federated graph* $\bar{G} = (G_j)_{j \in J}$ *is the union of the individual graphs, that is,* $\bar{G}^{fl} = \bigcup_{j \in J} G_j$. *The* flattening $\bar{\mathcal{G}}^{fl}$ *of an incomplete FDS* $\bar{\mathcal{G}} = (\bar{G}^a, G^i)$ *is the incomplete data source* $\bar{\mathcal{G}}^{fl} = ((\bar{G}^a)^{fl}, G^i)$ *whose available graph is the flattening of the available federated graph of* $\bar{\mathcal{G}}$. *The* flattening $\bar{\boldsymbol{C}}^{fl}$ *of a set* $\bar{\boldsymbol{C}}$ *of indexed completeness statements is the set* $\bar{\boldsymbol{C}}^{fl} = \{\, C \mid (C, k) \in \bar{\boldsymbol{C}} \}$, *where we ignore the indices.*

Note that the notion of federated entailment is different from the entailment between a set of completeness statements and a query defined in Section 4 in the sense that we now have to deal with indexed completeness statements.

Definition 18 (Federated Completeness and Entailment). *A federated query* $\bar{Q}$ *is* complete *over an incomplete FDS* $\bar{\mathcal{G}} = (\bar{G}^a, G^i)$, *written* $\bar{\mathcal{G}} \models_{fed} Compl(\bar{Q})$, *if* $[\![\bar{Q}]\!]_{(j_0, \bar{G}^a)} = [\![\bar{Q}^{fl}]\!]_{G^i}$ *for any IRI* $j_0 \in J$, *that is, the evaluation of* $\bar{Q}$ *over the available federated graph returns the same result as evaluating the flattening of* $\bar{Q}$ *over the ideal graph. If* $\bar{\boldsymbol{C}}$ *is an indexed set of completeness statements, then* $\bar{\mathbf{C}}$ entails $Compl(\bar{Q})$, *written* $\bar{\boldsymbol{C}} \models_{fed} Compl(\bar{Q})$, *if* $\bar{\mathcal{G}} \models_{fed} \bar{\boldsymbol{C}}$ *implies* $\bar{\mathcal{G}} \models_{fed} Compl(\bar{Q})$ *for all incomplete FDSs* $\bar{\mathcal{G}}$.

If Q is a basic query, then we say that Q is complete over $\bar{\mathcal{G}}$ if Q is complete over the flattening of $\bar{\mathcal{G}}$, that is, $\bar{\mathcal{G}} \models_{fed} Compl(Q)$ iff $\bar{\mathcal{G}}^{fl} \models Compl(Q)$. This means that Q is complete if evaluated over the *union* of all sources in the federation.

Proposition 19 (Completeness of Basic Queries). *Let* $\bar{\boldsymbol{C}}$ *be a set of indexed completeness statement and* Q *be a basic query. Then*

$$\bar{\boldsymbol{C}} \models_{fed} Compl(Q) \quad \textit{iff} \quad \bar{\boldsymbol{C}}^{fl} \models Compl(Q)$$

This means that we can check the completeness of a basic query with the criterion in Theorem 10 in Section 4.1. A federated query $\bar{Q}$ is a *federated version* of a basic query Q if $\bar{Q}^{fl} = Q$. In other words, by dropping the service calls from $\bar{Q}$ we obtain Q.

Theorem 20. ***(Smart Rewriting)***. *Let* $\bar{\boldsymbol{C}}$ *be a set of indexed completeness statement and* Q *be a basic query such that* $\bar{\boldsymbol{C}} \models_{fed} Compl(Q)$. *Then:*

1. *One can compute a federated version* $\bar{Q}$ *of* Q *such that* $\bar{\boldsymbol{C}} \models_{fed} Compl(\bar{Q})$.
2. *Moreover, whenever* $(\bar{G}^a, G^i) \models_{fed} \bar{\boldsymbol{C}}$, *then*

$$[\![Q]\!]_{\bigcup_{j \in J} G_j^a} \;=\; [\![\bar{Q}]\!]_{(j_0, \bar{G}^a)} \qquad \textit{for any } j_0 \in J.$$

To retrieve all answers for an arbitrary query, we have to evaluate each triple pattern over the union of all sources. For a complete query, the federated version evaluates each triple pattern only over a single source. Therefore, the evaluation of the federated version is in general much more efficient.

Example 21 ***(Federated Data Sources).*** Consider the two data sources shown in Fig. 3 plus an additional data source named FB (= Facebook) with the completeness statement $C_{fb} = Compl(\{(?m, likes, ?l)\} \mid \{(?m, type, Movie), (?m, director, tarantino)\})$ and the query: $Q_{fb} = (\{?m, ?l\}, \{(?m, type, Movie), (?m, director, tarantino), (?m, likes, ?l)\})$ that asks for the number of *likes* of Tarantino's movies.

In order to answer this query efficiently over the three data sources, whose endpoints are reachable at the IRIs `DBPe`, `LMDBe` and `FBe`, we compute a federated version $\bar{Q}_{fb}$. The completeness statements in Fig. 3 plus C_{fb} entail wrt. "$\models_{fed}$" the completeness of the query Q_{fb} (see Definition 18). By Theorem 20 we can compute a complete federated version $\bar{Q}_{fb}$, which in this case is $\bar{Q}_{fb} = (\{?m, ?l\}, \{$ `(SERVICE LMDBe` $\{(?m, type, Movie), (?m, director, tarantino)\}$`)}) AND (SERVICE FBe` $\{(?m, likes, ?l)\}$`)`, whose answer is complete.

6 Discussion

We now discuss some aspects underlying the completeness framework.

Availability of Completeness Statements. At the core of the proposed framework lies the availability of completeness statements. We have discussed in Section 2 how existing data sources like IMDb already incorporate such statements (Figure 1) and how they can be made machine-readable with our framework. The availability of completeness statements rests on the assumption that a domain "expert" has the necessary background knowledge to provide such statements.

We believe that it is in the interest of data providers to annotate their data sources with completeness statements in order to increase their value. Indeed, users can be more inclined to prefer data sources including "completeness marks" to other data sources. Moreover, in the era of crowdsourcing the availability of independent "ratings" from users regarding the completeness of data can also contribute (like in Wikipedia and OpenStreetMap), in a bottom up manner, to the description of the completeness of data sources. For instance, when looking up information about Stanley Kubrick in DBpedia, as a by-product users can provide feedback as to whether all of Kubrick's movies are present. One can also imagine approaches based on gamification.

Maintenance. If edits of a source are logged, log items could be automatically translated into updates of statements. For *non-authoritative* sources, temporal guards can be used; e.g., instead of saying "complete for all movies by Tarantino", one would say "complete for movies by Tarantino in 2010".

Complexity. All completeness checks presented in this paper are NP-complete. The hardness holds because classical conjunctive query containment can be encoded into completeness checking [17]; the NP upper bound follows because all completeness checks require conjunctive query evaluation at their core. In practice, we expect these checks to be fast, since queries and completeness statements are likely to be small. After all, this is the same complexity as the one of query evaluation and query optimization of basic queries, as implemented in practical database management systems.

Vocabulary Heterogeneity. In practice, a query may use a vocabulary different from that of some data sources. In this work, we assume the presence of a global schema. Indeed, one could use the `schema.org` vocabulary for queries, since it has already been mapped to other vocabularies (e.g., DBpedia).

The `CoRNER` Implementation. To show the feasibility of our proposal, we developed the `CoRNER` system. It implements the completeness entailment procedure for basic and `DISTINCT` queries with ρDF. The system is implemented in Java and uses the Apache Jena library. It is downloadable at `http://rdfcorner.wordpress.com`.

7 Concluding Remarks and Future Work

The availability of distributed and potentially overlapping RDF data sources calls for mechanisms to provide qualitative characterizations of their content. In this respect, we have identified *completeness* as one important dimension. The motivation underlying this work stems from the fact that although completeness information is present in some available data sources (e.g., IMDb discussed in Section 2) it is neither formally represented nor automatically processed. We have introduced a formal framework for the declarative specification of completeness statements about RDF data sources and underlined how the framework can complement existing initiatives like `VoID`. Then, we studied "how" completeness statements can be used in the problem of completeness of query answering. In this respect we considered queries over single and federated data sources and showed how to assess query completeness. We believe that our research can be the starting point of further investigation of the problem of completeness of information on the Web. Considering other application scenarios of completeness statements like data source integration and selection is in our research agenda.

References

1. Alexander, K., Cyganiak, R., Hausenblas, M., Zhao, J.: Describing linked datasets with the VoID vocabulary. Technical report, W3C (2011)
2. Arenas, M., Gutierrez, C., Pérez, J.: On the semantics of SPARQL. In: Semantic Web Information Management, pp. 281–307. Springer, Heidelberg (2010)
3. Brickley, D., Guha, R.V.: RDF vocabulary description language 1.0: RDF Schema. Technical report, W3C (2004)

4. Fürber, C., Hepp, M.: Using SPARQL and SPIN for data quality management on the Semantic Web. In: Abramowicz, W., Tolksdorf, R. (eds.) BIS 2010. LNBIP, vol. 47, pp. 35–46. Springer, Heidelberg (2010)
5. Harris, S., Seaborne, A.: SPARQL 1.1 query language. Technical report, W3C (2013)
6. Harth, A., Speiser, S.: On completeness classes for query evaluation on linked data. In: AAAI (2012)
7. Hartig, O., Bizer, C., Freytag, J.-C.: Executing SPARQL queries over the web of linked data. In: Bernstein, A., Karger, D.R., Heath, T., Feigenbaum, L., Maynard, D., Motta, E., Thirunarayan, K. (eds.) ISWC 2009. LNCS, vol. 5823, pp. 293–309. Springer, Heidelberg (2009)
8. Heath, T., Bizer, C.: Linked Data: Evolving the Web into a Global Data Space. Synthesis Lectures on the Semantic Web. Morgan & Claypool Publishers (2011)
9. Klyne, G., Carroll, J.J.: Resource Description Framework (RDF): Concepts and abstract syntax. Technical report, W3C (2004)
10. Letelier, A., Pérez, J., Pichler, R., Skritek, S.: Static analysis and optimization of semantic web queries. In: PODS, pp. 89–100 (2012)
11. Levy, A.Y.: Obtaining complete answers from incomplete databases. In: Proc. VLDB, pp. 402–412 (1996)
12. Motro, A.: Integrity = Validity + Completeness. ACM TODS 14(4), 480–502 (1989)
13. Muñoz, S., Pérez, J., Gutierrez, C.: Simple and efficient minimal RDFS. J. Web Sem. 7(3), 220–234 (2009)
14. Pérez, J., Arenas, M., Gutierrez, C.: Semantics and complexity of SPARQL. ACM TODS 34(3), 16 (2009)
15. Picalausa, F., Vansummeren, S.: What are real SPARQL queries like? In: SWIM (2011)
16. Polleres, A., Feier, C., Harth, A.: Rules with contextually scoped negation. In: Sure, Y., Domingue, J. (eds.) ESWC 2006. LNCS, vol. 4011, pp. 332–347. Springer, Heidelberg (2006)
17. Razniewski, S., Nutt, W.: Completeness of queries over incomplete databases. PVLDB 4(11), 749–760 (2011)
18. Seaborne, A., Polleres, A., Feigenbaum, L., Williams, G.T.: SPARQL 1.1 federated query. Technical report, W3C (2013)
19. Hamilton, H.J., Wang, X., Bither, Y.: An ontology-based approach to data cleaning. Department of Computer Science, University of Regina (2005)

Empirical Study of Logic-Based Modules: Cheap Is Cheerful

Chiara Del Vescovo[1], Pavel Klinov[2], Bijan Parsia[1], Ulrike Sattler[1], Thomas Schneider[3], and Dmitry Tsarkov[1]

[1] University of Manchester, UK
{delvescc,bparsia,sattler,tsarkov}@cs.man.ac.uk
[2] University of Ulm, Germany
pavel.klinov@uni-ulm.de
[3] Universität Bremen, Germany
tschneider@informatik.uni-bremen.de

Abstract. For ontology reuse and integration, a number of approaches have been devised that aim at identifying modules, i.e., suitably small sets of "relevant" axioms from ontologies. Here we consider three logically sound notions of modules: MEX modules, only applicable to inexpressive ontologies; modules based on semantic locality, a sound approximation of the first; and modules based on syntactic locality, a sound approximation of the second (and thus the first), widely used since these modules can be extracted from OWL DL ontologies in time polynomial in the size of the ontology.

In this paper we investigate the quality of both approximations over a large corpus of ontologies, using our own implementation of semantic locality, which is the first to our knowledge. In particular, we show with statistical significance that, in most cases, there is no difference between the two module notions based on locality; where they differ, the additional axioms can either be easily ruled out or their number is relatively small. We classify the axioms that explain the rare differences into four kinds of "culprits" and discuss which of those can be avoided by extending the definition of syntactic locality. Finally, we show that differences between MEX and locality-based modules occur for a minority of ontologies from our corpus and largely affect (approximations of) expressive ontologies – this conclusion relies on a much larger and more diverse sample than existing comparisons between MEX and syntactic locality-based modules.

1 Introduction

Some notable examples of ontologies describe large and loosely connected domains, as it is the case for SNOMED CT, the Systematized Nomenclature Of MEDicine, Clinical Terms,[1] which describes the terminology used in medicine including diseases, drugs, etc. Users often are not interested in a whole ontology

[1] http://www.ihtsdo.org/snomed-ct/

H. Alani et al. (Eds.): ISWC 2013, Part I, LNCS 8218, pp. 84–100, 2013.

$\mathcal{O}$ but rather only in a limited part of it which is relevant to their application. One recently explored technique for addressing this situation is to use *modules*, i.e., suitably small subsets of $\mathcal{O}$ that behave for specific purposes like the original ontology over a given *signature* Σ, i.e., a set of terms (classes and properties).

Using a module rather than a whole ontology aims at improving performance since only information that is relevant to a restricted vocabulary is processed. However, the correctness of the outcome can be guaranteed only if the used modules satisfy certain well-defined properties. For example, reasoning-based tasks require the modules to *provide coverage* for $\mathcal{O}$ over Σ, i.e., preserve *all* the entailments of $\mathcal{O}$ over Σ (they are called *logical modules* [9,4]). Applications of logical modules include reuse of (a part of) well-established ontologies, ontology integration, and computing justifications to debug ontologies [11]. In these scenarios, though, a stronger notion of logical module is required that satisfies also two additional properties [15,19]: *self-containment* and *depletion*. The former means that the module preserves entailments over all terms that occur in the module (not just those used to extract the module). The latter means that $\mathcal{O} \setminus \mathcal{M}$ does not entail any non-tautological axioms over Σ. In this paper we will analyze only depleting and self-contained logical modules.

Interestingly, a *minimal* depleting and self-contained module for a signature Σ is, under some mild conditions, uniquely determined [15]. Extracting such modules is, unfortunately, computationally hard or even undecidable for expressive ontology languages [10,17,18]. In order to identify notions of modules whose extraction is feasible we can follow two alternative strategies. The first one consists of restricting the expressivity of the ontology language, as in the case of the MEX approach [14]: the MEX system allows for the extraction in polynomial time of the minimal self-contained and depleting module from acyclic $\mathcal{ELI}$ terminologies. The second strategy consists of looking for practical sufficient conditions to guarantee the properties of logical modules without imposing minimality on the module $\mathcal{M}$, as it is the case for the family of logical modules known as *locality-based modules* (*LBMs*) [3]; these modules can be extracted from ontologies as expressive as $\mathcal{SROIQ}$, are self-contained and depleting, but can contain axioms that are not relevant to preserve any entailment over the given Σ.

The family of LBMs consists of module notions that are parameterized according to two features: (1) the technique used for identifying which axioms need to be included in the module (*semantic* or *syntactic*); (2) the kind of placeholder(s) used for those terms not included in the signature (*bottom*, *top*, or *nested*). In the next two paragraphs we provide an intuitive discussion of the meaning of these two features.

The extraction of semantic LBMs requires entailment checks against an empty ontology and thus involve reasoning, which makes the computation as hard as reasoning. Moreover, the kind of reasoning service used is rather unusual for DL reasoners.[2] Hence, although algorithms for extracting semantic LBMs are known, until now and to the best of our knowledge they had not been implemented.

[2] DL reasoners usually *classify* an ontology: test it for consistency and all concept names for satisfiability/mutual subsumption.

In contrast, the extraction of syntactic LBMs involves *only* parsing the axioms of the ontology. Algorithms for the extraction of syntactic LBMs are known that run in time polynomial in the size of the ontology (thus much cheaper than reasoning), and are implemented in the OWL API.[3]

The kind of placeholder(s) used for semantic and syntactic LBMs gives a flavour of the different module notions. The bottom variants of LBMs provide a view of $\mathcal{O}$ from Σ "upwards" since they contain all named superclasses of class names in Σ; the top variants instead provide a view of $\mathcal{O}$ from Σ "downwards" since they contain all named subclasses of class names in Σ; finally, the nested variants provide a view of $\mathcal{O}$ "within" Σ since they still provide coverage for Σ as the other variants, but they do not necessarily contain all the sub- or super-classes of the classes in Σ.

This paper empirically studies the seven module notions depicted in Fig. 1 which summarizes their notations and their inclusion relations. Each node represents a module notion; the one for the MEX module is shadowed because this method can be used only for $\mathcal{ELI}$ acyclic ontologies. The MEX notion is in the same column as the nested versions because MEX modules provide a similar view of $\mathcal{O}$ "within" Σ.

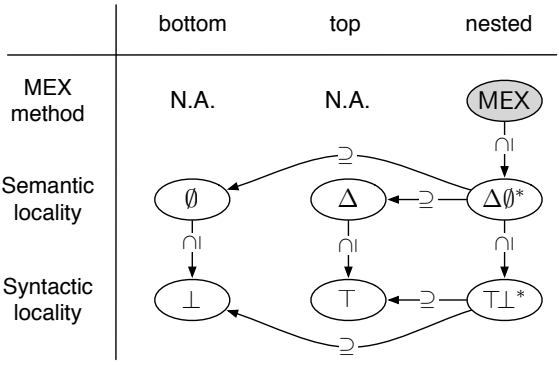


Fig. 1. Inclusion relations between the 7 notions of modules investigated

As shown in Fig. 1, the MEX module for a signature Σ is a subset of the nested semantic LBM, and for each variant bottom, top, and nested, the semantic LBMs are contained in the corresponding syntactic ones. Hence, syntactic locality can be seen as an approximation of semantic locality which, in turn, is an approximation of MEX modules. This gives rise to the question of how *good* these approximations are: how much larger are the modules extracted by the approximations, and how much faster is the extraction?

This paper provides emprical answers to these questions by comparing different modules systematically extracted from a large corpus of real-life ontologies. Specifically, semantic LBMs are compared with syntatic LBMs and with MEX modules (for acyclic $\mathcal{ELI}$ ontologies). This paper substantially extends

[3] `http://owlapi.sourceforge.net/`

the previous experiments reported in [14] where MEX modules were compared with syntactic bottom modules on a sample of 5000 random signatures and the SNOMED CT ontology. We perform our study on a larger corpus (not restricted to $\mathcal{ELI}$), compare more notions of logical modules, and also provide rigorous statistical significance results.

The main contributions of this paper are summarized as follows:

- We show with statistical significance that, for almost all members of a large corpus of existing ontologies, there is no difference between any syntactic LBM and its semantic counterpart. In the few cases where differences occur, those are extremely modest so that it is questionable whether extracting semantic LBMs is worth the increased computational cost.
- We isolate four *culprits*, i.e., patterns of axioms that completely explain those rare differences. One includes simple tautologies that can be removed in a straightforward preprocessing step.
- Our results show that the extraction of semantic LBMs, which is in principle hard, is feasible in practice: on average, it is between 3 times (for top-modules) and 15 times (for bottom- and nested-modules) slower than the extraction of syntactic LBMs, and both only take milliseconds to seconds for most ontologies below 10K axioms.
- To obtain these results, we use our own implementation of semantic locality which, to the best of our knowledge, is the first ever to be implemented.
- We modify the original corpus to obtain for each ontology an acyclic $\mathcal{EL}$ version suitable for the use with the MEX system. We then compare MEX-modules and the nested-variants of LBMs, and find differences in only ∼27% of the corpus. We explain one reason for the largest differences observed.

2 Preliminaries

We assume the reader to be familiar with Description Logic languages (e.g. $\mathcal{SROIQ}$ [1,13]), and aim here at fixing the notations and at defining the key notions around module extraction, with a focus on locality-based modules [3] and MEX modules [14].

Let $\mathcal{O}$ denote an ontology, $\mathsf{N_C}$ a set of class names, and $\mathsf{N_R}$ a set of property names. A *signature* is a set $\Sigma \subseteq \mathsf{N_C} \cup \mathsf{N_R}$ of *terms*. Given a class, property, or axiom X, we call the set of terms in X the *signature of* X, denoted $\widetilde{X}$. Given a $\mathcal{SROIQ}$ ontology $\mathcal{O}$, a set $\mathcal{M} \subseteq \mathcal{O}$ of axioms from $\mathcal{O}$, and a signature Σ, we say that $\mathcal{O}$ is a *deductive Σ-conservative extension* (Σ-*dCE*) of $\mathcal{M}$ if, for all $\mathcal{SROIQ}$-axioms α with $\widetilde{\alpha} \subseteq \Sigma$, it holds that $\mathcal{O} \models \alpha$ if and only if $\mathcal{M} \models \alpha$. $\mathcal{O}$ is a *model Σ-conservative extension* (Σ-*mCE*) of $\mathcal{M}$ if $\{\mathcal{I}|_\Sigma \,|\, \mathcal{I} \models \mathcal{O}\} = \{\mathcal{I}|_\Sigma \,|\, \mathcal{I} \models \mathcal{M}\}$. Dually, $\mathcal{M}$ is a *dCE-based module* of $\mathcal{O}$ for Σ if $\mathcal{O}$ is a Σ-dCE of $\mathcal{M}$, and it is an *mCE-based module for* Σ if $\mathcal{O}$ is a Σ-mCE of $\mathcal{M}$. All dCE-based modules are also mCE-based modules, whilst the converse is not always true. A module $\mathcal{M} \subseteq \mathcal{O}$ for Σ is called *depleting* if there is no non trivial entailment η over Σ such that $\mathcal{O} \setminus \mathcal{M} \models \eta$; $\mathcal{M}$ is called *self-contained* if $\mathcal{M}$ is a module for $\Sigma = \widetilde{\mathcal{M}}$.

Since $\mathcal{M} \subseteq \mathcal{O}$ the monotonicity of $\mathcal{SROIQ}$ implies that every entailment η over Σ derivable from $\mathcal{M}$ is also derivable from $\mathcal{O}$. Deciding the converse

direction is in general computationally hard, or even undecidable for expressive DLs [10,17,18]. Since we do not need to find *all* the subsets of $\mathcal{O}$ that are a module for Σ, we can use easier conditions which guarantee that a set of axioms $\mathcal{M} \subseteq \mathcal{O}$ is a module for Σ.

Let Σ be a signature and $\mathcal{O}$ be an ontology. Let $x \in \{\mathsf{MEX}, \emptyset, \Delta, \bot, \top\}$ be a notion of module. For each such notion, an oracle "x-check" can be defined that determines whether an axiom α may be involved in preserving an entailment η of $\mathcal{O}$ over Σ. Then, the x-module $x\text{-mod}(\Sigma, \mathcal{O})$ for Σ in $\mathcal{O}$ can be computed by performing Algorithm 1.

Algorithm 1. Extraction of an x-module for Σ

Input: Ontology $\mathcal{O}$, seed signature Σ, oracle x-check
Output: x-module $\mathcal{M}$ of $\mathcal{O}$ w.r.t. Σ

$\mathcal{M} \leftarrow \emptyset$; $\mathcal{O}' \leftarrow \mathcal{O}$
repeat
  changed $\leftarrow$ `false`
  for all $\alpha \in \mathcal{O}'$ **do**
    if the x-check for α against $\Sigma \cup \widetilde{\mathcal{M}}$ is positive **then**
      $\mathcal{M} \leftarrow \mathcal{M} \cup \{\alpha\}$; $\mathcal{O}' \leftarrow \mathcal{O}' \setminus \{\alpha\}$; changed $\leftarrow$ `true`
until changed = `false`
return $\mathcal{M}$

Algorithm 1 is a special case of the one in [3, Figure 4], and its output $\mathcal{M}$ does not depend on the order in which the axioms α are selected [3].

Due to space limitations, we can just briefly sketch the intuition behind the definition of each oracle and the corresponding results of interest for this paper. We refer the interested reader to [3,14] for further details.

The MEX System. In [14], the notion of a MEX-module is defined for acyclic terminologies, i.e., ontologies that satisfy two conditions: (1) they only contain axioms of the form $\mathtt{A} \equiv \mathtt{C}$ or $\mathtt{A} \sqsubseteq \mathtt{C}$ where $\mathtt{A}$ is a class name and $\mathtt{C}$ is a complex class; (2) for each $\mathtt{A}$, there is at most one axiom with $\mathtt{A}$ on the left-hand side; if one such axiom α exists, then $\mathtt{A}$ is said to be *defined*, and to be *directly dependent on* all the terms $\mathtt{X}$ that occur on the right-hand side of α (denoted $\mathtt{A} \succ \mathtt{X}$). The MEX method requires to determine for each defined class $\mathtt{A}$ the set $\mathsf{depend}_{\mathcal{O}}(\mathtt{A})$ of all the terms $\mathtt{X}$ in $\mathcal{O}$ such that the pair $(\mathtt{A}, \mathtt{X})$ belongs to the transitive closure of $\succ$. Intuitively, then, the MEX-check for an axiom α against a signature Σ tests whether either α defines a class $\mathtt{A} \in \Sigma \cup \widetilde{\mathcal{M}}$ and *uses*[4] at least one term $\mathtt{X} \in \mathsf{depend}_{\mathcal{O}}(\mathtt{A}) \cap (\Sigma \cup \widetilde{\mathcal{M}})$ in $\mathcal{O} \setminus \mathcal{M}$, or if every term on which $\mathtt{A}$ depends only via $\equiv$-axioms is *used to define*[4] some term in $\Sigma \cup \widetilde{\mathcal{M}}$. The authors prove that, if $\mathcal{O}$ is an acyclic $\mathcal{ELI}$ ontology, then using the oracle MEX-check in Algorithm 1 generates the minimal depleting self-contained module for a signature Σ in polynomial time.

Semantic Locality. In [3], the authors define a family of notions of locality with different parameters, the prominent notions being those where the placeholder

[4] The expressions *use* and *used to define* are high-level intuitive descriptions of the two conditions given in [14, Fig. 4], to which we refer the reader since a formal definition goes beyond the scope of this paper.

x belongs to $\{\emptyset, \Delta\}$. These two notions of locality can be intuitively described as follows: a $\mathcal{SROIQ}$ axiom α is $\emptyset$-*local* (resp. Δ-*local*) w.r.t. signature Σ if α' obtained by replacing all terms in $\widetilde{\alpha} \setminus \Sigma$ with $\bot$ (resp. $\top$) is a tautology, in which case the x-check returns negative. This treatment of α independently of the remaining axioms distinguishes the $\emptyset$- and Δ-check (as well as the $\bot$- and $\top$-check introduced in the next paragraph) from the MEX-check; hence the name *local*. The authors of [3] prove that, if all axioms in $\mathcal{O} \setminus \mathcal{M}$ are $\emptyset$-local (or all axioms are Δ-local) w.r.t. $\Sigma \cup \widetilde{\mathcal{M}}$, then $\mathcal{M}$ is an mCE-based (and hence dCE-based) module of $\mathcal{O}$ for Σ. Since deciding $\emptyset$- or Δ-locality requires tautology checks, this problem is as hard as standard reasoning. In some cases, α' is not a $\mathcal{SROIQ}$ axiom, so standard reasoners need to be extended.

Syntactic Locality. In order to achieve *tractable* module extraction, the two syntactic notions of x-locality for $x \in \{\bot, \top\}$ have been defined in [3]. Similarly to semantic locality, the x-check for an axiom α against a signature Σ operates on the transformed axiom α' obtained by replacing all terms not in Σ with the placeholder x. However, rather than invoking a reasoner, the x-check of α against Σ makes use of a simple syntactic test [3, Sec. 5.5]. For example, $\bot \sqsubseteq \mathtt{C}$ is clearly a tautology for each class $\mathtt{C}$. If the x-check is negative, α is said to be $\bot$- or $\top$-*local* w.r.t. Σ. The x-check used in syntactic LBMs is sound in identifying non-tautological axioms, but it may fail to spot a tautology, i.e., every $\emptyset$-local (Δ-local, resp.) axiom w.r.t. Σ is also $\bot$-local ($\top$-local, resp.) w.r.t. Σ, but not vice versa. Thus, also $\bot$- and $\top$-modules are mCE- and dCE-based modules for Σ. Applying the syntactic rules requires polynomial time, hence the extraction of this kind of modules is performed in time polynomial in the size of the ontology.

Modules based on syntactic (semantic) locality can be made smaller by iteratively nesting $\top$- and $\bot$-extraction (Δ- and $\emptyset$-extraction), again obtaining mCE- and dCE-based modules [3,19], called $\top\bot^*$- and $\Delta\emptyset^*$-*modules*.

Algorithm 1 guarantees that the module notions considered here are self-contained and depleting: self-containment holds because of the iteration until the signature of $\mathcal{M}$ remains unchanged; depletion holds because the axioms left out of $\mathcal{M}$ are those whose x-check against the enlarged signature is negative.

3 Research Questions and Experimental Design

A natural question arising is whether syntactic and semantic LBMs differ in practice, and, if yes, by how much. A second question is whether semantic module extraction is *noticeably* more costly: the x-check has to be carried out often— once per axiom and signature that the algorithm goes through— and it is hard to predict the feasibility of semantic LBM extraction. Altogether, we want to know whether syntactic LBMs are a *good* approximation of semantic LBMs, and how much they differ in cost. Similarly, for acyclic $\mathcal{ELI}$ ontologies the analogous question arises: how good an approximation of MEX modules are LBMs?

An answer to these questions will allow for a more informed choice of which module extraction technique to select. One can always construct ontologies with huge differences in size and time between syntactic and semantic LBMs and between LBMs and MEX modules. Here, we are interested in these differences

in currently available ontologies, and thus we need to design, run, and analyse suitable experiments.

Selection of the Corpus. For our experiments, we have built a corpus containing: (1) all the ontologies from the NCBO BioPortal ontology repository,[5] version of November 2012; (2) ontologies from the TONES repository[6] which have already been studied in previous work on modularity [6]: Koala, Mereology, University, People, miniTambis, OWL-S, Tambis, Galen. From this corpus, we have removed ontologies that cannot be downloaded, whose .owl file is corrupted or impossible to parse, or which are inconsistent. Furthermore, we have excluded those large ontologies (exceeding 10K axioms) where the extraction of a semantic LBM repeatedly took more than 2 minutes: for each such ontology, the estimated time needed to perform our experiments could have exceeded 300 hours. However, to include at least one case of a huge ontology, we have kept in the corpus NCI, an $\mathcal{SH}(\mathcal{D})$ ontology with 123,270 axioms.

This selection results in a corpus of 242 ontologies, which even beside NCI greatly vary in expressivity (from $\mathcal{AL}$ to $\mathcal{SROIQ}(\mathcal{D})$) and in size (10–16,066 axioms, 10–16,068 terms) [12]. For a full list of the corpus, please refer to [5].

As mentioned above, it is not possible for some ontologies to test Δ-locality (and thus for extracting Δ- and $\Delta\emptyset^*$-modules) using standard DL reasoners, see [5] for details. To cover these cases, we have extended the reasoner FaCT++ to cover the use of the $\top$-role as required by the semantic locality tests.

Since MEX handles only acyclic $\mathcal{ELI}$ ontologies, we created an $\mathcal{ELI}$ version $\mathcal{ELI}(\mathcal{O})$ of each ontology $\mathcal{O}$ in our corpus by filtering unsupported axioms and breaking terminological cycles. A principled way of doing this is beyond the scope of this paper, and we have used the heuristic described in [5]. The resulting corpus contains 239 ontologies since 3 were left empty after the $\mathcal{ELI}$-fication.

Comparing Modules and Locality. In order to compare syntactic and semantic locality, as well as LBMs and MEX modules, we want to understand (1) whether, for a given seed signature Σ, it is likely that there is a difference between the syntactic, the semantic, and the MEX modules for Σ; if so, the size of the difference;[7] and (2) how feasible the extraction of semantic LBMs is.

For this purpose, we compare (a) $\emptyset$-semantic and $\bot$-syntactic locality, Δ-semantic and $\top$-syntactic locality, (b) $\emptyset$- and $\bot$-modules, Δ- and $\top$-modules, $\Delta\emptyset^*$- and $\top\bot^*$-modules, (c) MEX modules and $\Delta\emptyset^*$-modules.

Due to the recursive nature of Algorithm 1, our investigation is both on a

per-axiom-basis: given axiom α and signature Σ, is it likely that α is $\emptyset$-local (Δ-local, resp.) w.r.t. Σ but not $\bot$-local ($\top$-local, resp.) w.r.t. Σ?

per-module basis: given a signature Σ, is it likely that

- $\bot\text{-mod}(\Sigma, \mathcal{O}) \neq \emptyset\text{-mod}(\Sigma, \mathcal{O})$, or
- $\top\text{-mod}(\Sigma, \mathcal{O}) \neq \Delta\text{-mod}(\Sigma, \mathcal{O})$, or

[5] http://bioportal.bioontology.org

[6] http://owl.cs.manchester.ac.uk/repository/

[7] Recall: the MEX module is always a subset of the semantic Σ-module, which is always a subset of the syntactic Σ-module.

– $\top\bot^*\text{-mod}(\Sigma, \mathcal{O}) \neq \mathit{\Delta}\emptyset^*\text{-mod}(\Sigma, \mathcal{O})$, or
– $\mathit{\Delta}\emptyset^*\text{-mod}(\Sigma, \mathcal{O}) \neq \mathsf{MEX}\text{-mod}(\Sigma, \mathcal{O})$?

If yes, is it likely that the difference is large?

Clearly we need to pick, for each ontology in our corpus, a suitable set of signatures, and this poses a significant problem. A full investigation is infeasible: if $m = \#\widetilde{\mathcal{O}}$, there are 2^m possible seed signatures, so that testing axioms for locality against *all* the signatures is already impossible for $m \sim 100$. One could assume that comparing modules is easier since many signatures can lead to the same module. However, previous work [6,8] has shown that the number of modules in ontologies is, in general, exponential w.r.t. the size of the ontology. Still, different seed signatures can lead to the same module, which makes it hard to extract enough *different* modules.

We will consider seed signatures of two kinds: genuine seed signatures and random seed signatures.

Genuine Seed Signatures. A module does not necessarily show an internal coherence: e.g., if we had an ontology $\mathcal{O}$ about the domains of geology and philosophy, we could extract the module for the signature $\Sigma = \{\texttt{Epistemology}, \texttt{Mineral}\}$. That module is likely to be the union of the two disjoint modules for $\Sigma_1 = \{\texttt{Epistemology}\}$ and $\Sigma_2 = \{\texttt{Mineral}\}$ [7].

In contrast, *genuine modules* can be said to be coherent: they are those modules that cannot be decomposed into the union of two "$\subseteq$"-uncomparable modules. Interestingly, a module $\mathcal{M}$ is genuine iff there exists an axiom α such that $\mathcal{M} = x\text{-mod}(\widetilde{\alpha}, \mathcal{O})$. As a consequence, there are only linearly many genuine modules in the size of $\mathcal{O}$, and extracting one module per axiom is enough for obtaining all of them. Moreover, all modules of $\mathcal{O}$ are composed from genuine modules [7]. Thus, genuine modules are of special interest, and we can investigate *all* of them, together with the corresponding *genuine signatures*.

Random Seed Signatures. Since a full investigation of all the signatures is impossible, we compare locality—both on a per-axiom and per-module basis—as well as LBMs and MEX modules on a *random* signature Σ, which we select by setting each named entity `E` in the ontology to have probability $p = 1/2$ of being included in Σ. This ensures that each Σ will have the same probability to be chosen. This approach has a clear setback: the random variable "size of the seed signature generated" follows a binomial distribution, so a random seed signature is highly likely to be rather large and to contain half the terms of the ontology. However, we do not yet have enough insight into what *typical* seed signatures are for module extraction, so biasing the selection of signatures to, for example, those of a certain size has no *rationale*. In contrast, selecting random seed signatures avoids the introduction of any bias. Moreover, this choice is complementary to the selection of *all* the genuine signatures, which are in general small.

With this in mind, we will analyze the modules obtained by random signatures with $p = 1/2$, and we will see in Section 4 that the module sizes obtained do allow for a reliable statement about the differences observed.

How many seed signatures do we have to sample from a given ontology $\mathcal{O}$ in order to obtain statistically significant statements about modules determined

by the real population of *all* signatures from $\mathcal{O}$? We apply the usual statistical model of confidence intervals [20], aiming at a confidence level of 95% that the true proportion of differences between modules – i.e., the proportion of seed signatures that lead to different modules – lies in the confidence interval ($\pm 5\%$) of the observed proportion. Then we can generalize the conclusions for the random sample to the full population because the probability that the proportion of differences among modules for all seed signatures differs by no more than 5% from the proportion observed in the sample (and reported in Section 4) is 95%. In order to reach this confidence level, we need a sample size of at least 385 elements, independently of the size of the full population: for a two-sided test to detect a change in the proportion defective of size δ in either direction, the minimum sample size is

$$N \geqslant \frac{p(1-p)}{\delta^2} z^2_{1-\alpha/2}\,,$$

where p is the observed proportion, α the significance level, and $z_{1-\alpha/2}$ the critical value of the underlying distribution [2]. Here, we use the normal distribution as an approximation of the binomial distribution which is usually assumed for proportions in random sampling; hence the significance level of $\alpha = 0.05$ leads to $z_{1-\alpha/2} \approx 1.96$. Furthermore, although we do not know the value p in advance, it is clear that $p(1-p) \leqslant 0.25$ because $0 \leqslant p \leqslant 1$. The confidence interval of $\pm 5\%$ determines the error of $\delta = 0.05$. Therefore, we obtain

$$N \geqslant \frac{0.25}{0.05^2} \cdot 1.96^2 \approx 384.16,$$

that is, a representative sample for these parameters needs at least 385 elements, and this number is independent of the population size. For ontologies with at least 9 elements in the signature, we will therefore draw a sample of size 400. For all other ontologies, we will look at *all* of the $\leqslant 400$ signatures.

Summary. We compare, for every ontology $\mathcal{O}$ in our corpus,

(T1) for random seed signatures Σ from $\mathcal{O}$,
- **(a)** for each axiom α in $\mathcal{O}$, is α
  - $\emptyset$-local w.r.t. Σ but not $\bot$-local w.r.t. Σ?
  - Δ-local w.r.t. Σ but not $\top$-local w.r.t. Σ?
- **(b)** is
  - $\bot$-mod$(\Sigma, \mathcal{O}) \neq \emptyset$-mod$(\Sigma, \mathcal{O})$?
  - $\top$-mod$(\Sigma, \mathcal{O}) \neq \Delta$-mod$(\Sigma, \mathcal{O})$?
  - $\top\bot^*$-mod$(\Sigma, \mathcal{O}) \neq \Delta\emptyset^*$-mod$(\Sigma, \mathcal{O})$?
  - $\Delta\emptyset^*$-mod$(\Sigma, \mathcal{ELI}(\mathcal{O})) \neq$ MEX-mod$(\Sigma, \mathcal{ELI}(\mathcal{O}))$?

(T2) the same questions **(a)** and **(b)**, with Σ ranging over *all* the genuine signatures $\tilde{\beta}$ for $\beta \in \mathcal{O}$.

Our sample selection includes large as well as small seed signatures: the random seed signatures created to answer T1 will tend to contain around half the terms in the ontology, while the signatures used to answer T2 will range over *all* signatures of *single axioms* and therefore tend to be small.

4 Results of the Experiments

4.1 Semantic Versus Syntactic Locality

No Differences in Locality. The main result of the experiment is that, for the vast majority of the ontologies in our corpus, no difference between syntactic and semantic locality is observed, for all three variants $\bot$ vs. $\emptyset$, $\top$ vs. Δ, and $\top\bot^*$ vs. $\Delta\emptyset^*$. More precisely, for 209 out of 242 ontologies, we obtain that:

(T1) for random seed signatures, there is no statistically significant difference
- **(a)** between semantic and syntactic locality of any kind,
- **(b)** between semantic and syntactic LBMs of any kind;

(T2) given *any* genuine signature, there is no such difference.

More specifically, for all randomly generated seed signatures and *all* genuine signatures, the corresponding bottom-modules (and the corresponding top- and nested-modules, respectively) agree, and every axiom is either $\bot$- and $\emptyset$-local, or none of both (and either $\top$- and Δ-local, or none of both).

The 209 ontologies include Galen and People, which are renowned for having unusually large $\bot$-modules [3,8].

In most cases, extracting a semantic and syntactic LBM each took only a few milliseconds, so a performance comparison is not meaningful. For some ontologies, the semantic LBM took considerably longer to extract than the syntactic: up to 5 times for nested-modules in Molecule Role, and up to 34 times in Galen.

Differences in Locality. We have observed differences between syntactic and semantic locality for 33 ontologies in our corpus. We call the axioms that cause these differences *culprits* – patterns of axioms which are not $\bot$-local ($\top$-local, respectively) w.r.t. some signature Σ, but which are $\emptyset$-local (Δ-local, respectively) w.r.t. Σ. We have identified four types of patterns, *a*–*d*, and we describe them in the following. Sometimes, culprit axioms *pull* additional axioms into the syntactic LBM, due to signature extension during module extraction.

We denote class *names* by $\mathtt{A}, \mathtt{B}$, complex classes by $\mathtt{C}, \mathtt{D}$, properties by $\mathtt{r}, \mathtt{s}, \ldots$, nominals by a, non-empty data ranges (e.g., `int` or $\mathtt{int}^{0..9}$) by R, possibly with indices. Σ denotes a signature for which a module is extracted or against which an axiom is checked for locality. Terms outside Σ are overlined; we further use notation $\mathtt{C}^{\bot}$ and $\mathtt{C}^{\top}$ to denote classes that are bottom- or top-equivalent due to the grammar defining syntactic locality in [3, Fig. 3] and the analogous grammar for semantic locality.

Culprits of Type *a* are simple tautologies that accidentally entered the "inferred view" (closure under certain entailments) of an ontology. These axioms do not occur in the original "asserted" versions and could, in principle, be detected in a simple preprocessing step. Type-*a* culprits occur in 10 ontologies of the above 33 and are of the kinds $\mathtt{A} \sqsubseteq \mathtt{A}$ or $\mathtt{r} \equiv (\mathtt{r}^-)^-$. Each such tautology is trivially $\emptyset$-local and Δ-local w.r.t. any Σ, but not always $\bot$- or $\top$-local: if Σ contains all terms in that tautology, then both sides of the subsumption (equivalence) are neither $\bot$- nor $\top$-equivalent.

Differences Caused Not Solely by Culprits of Type *a* have been observed for 27 ontologies. In only 6 of these cases, the differences affect modules; in the remaining 20, they only affect locality of single axioms (tests T1 a and T2 a). We will focus on the former 6, listed in Table 1, and refer to [5] for details on all 27.

Table 1. Ontologies that exhibit differences in modules

Ontology	Abbreviation	DL expressivity	#axioms	#terms
MiniTambis-repaired	MiniT	$\mathcal{ALCN}$	170	226
Tambis-full	Tambis	$\mathcal{SHIN}(\mathcal{D})$	592	496
Bleeding History Phenotype	BHO	$\mathcal{ALCIF}(\mathcal{D})$	1,925	581
Neuro Behavior Ontology	NBO	$\mathcal{AL}$	1,314	970
Pharmacogenomic Relationsh...	PhaRe	$\mathcal{ALCHIF}(\mathcal{D})$	459	311
Terminological and Ontological...	TOK	$\mathcal{SRIQ}(\mathcal{D})$	466	330

According to Table 1, differences between modules occur for ontologies of medium to large size and medium to high expressivity. Differences in locality alone additionally affect small ontologies such as Koala (42 axioms) and Pilot Ontology (85 axioms), as well as large ontologies such as Galen (4,735 axioms) and Experimental Factor Ontology (7,156 axioms). The number of axioms *causing* these differences (i.e., matching the culprit patterns) in the affected ontologies is small except for Galen, and most of the observed differences are relatively small.

Table 2 gives a representative selection of the differences in *modules* observed, plus the relative sizes of modules extracted for (T1) and (T2). For a complete overview, including differences in locality of single axioms, see the table in [5].

Table 2. Overview of observed differences between modules

Ontol.	Types affected	#diffs	size of diffs #axs	size of diffs (rel.)	size of $\Delta\emptyset^*$-modules (%) T1 range	T1 avg.	T2 range	T2 avg.	culprit type + freq.
miniT	bot, nested	14–25%	1–7	0–600%[b]	48–79	66	0–8	2	*c* 3
Tambis	bot, nested	32–57%	2–41[c]	1–62%[c]	75–88	82	0–34	9	*c* 8
BHO[a]	nested	17%	1–12	0–300%	55–72	65	0–31	4	*b* 31
NBO[a]	nested	3%	2	0–200%	64–78	71	0–3	0	*d* 3
PhaRe[a]	top, nested	1–8%	1–326[d]	0–6,520%[d]	50–70	60	0–8	1	*d* 10
TOK	top, nested	49–100%	1–7	0–9%	48–68	59	9–17	10	*d* 3

[a]differences only for genuine modules
[b]differences $> 5\%$ only for genuine modules
[c]differences > 11 axioms ($> 2\%$) only for genuine modules
[d]differences > 13 axioms ($> 1{,}300\%$) only for top-modules

The columns show: ontology name (abbreviations: see Table 1); type of modules affected; relative number of module pairs with differences; number of axioms in the differences (absolute and relative to the $\emptyset$- or Δ- or $\Delta\emptyset^*$-case); type of culprit present and number of axioms of this type involved in differences.

Table 2 shows small absolute differences for miniT, BHO, NBO, and TOK. In Tambis, large differences occur only for genuine modules. Finally, in PhaRe, large differences occur only for top-modules.

For all these ontologies, a single syntactic or semantic module was extracted within only a few milliseconds, making module extraction times roughly equal.

Culprits of Type *b* are axioms with an $\exists$-restriction on a set of nominals or a non-empty data range on the right-hand side, such as $\mathtt{A} \sqsubseteq \exists \overline{\mathtt{r}}.\{a_1, \ldots, a_n\}$ or $\mathtt{A} \sqsubseteq \exists \overline{\mathtt{r}}.R$. These axioms are Δ-local w.r.t. a signature that does not contain $\mathtt{r}$ because they become tautologies if r is replaced by $\top$. However, they can never be $\top$-local unless $\mathtt{A}$ is replaced by some $\mathtt{C}^{\bot}$.

Culprit-b axioms affect genuine modules of BHO, and (only) locality of single axioms for 4 more ontologies. We observed a variant $\mathtt{A} \equiv \mathtt{C}^{\top} \sqcap \exists \overline{\mathtt{r}}.R$.

Culprits of Type *c* are axioms α that contain a class description $\mathtt{C}$ such that (a) $\mathtt{C}$ becomes equivalent to $\bot$ (or $\top$) if all terms outside Σ are replaced by $\bot$ (or $\top$); (b) this causes α to be semantically $\bot$-local (or $\top$-local); but (c) the grammars for syntactic locality do not "detect" $\mathtt{C}$ to be a $\mathtt{C}^{\bot}$ (or $\mathtt{C}^{\top}$). For example, $\mathtt{C} = \forall \mathtt{r}.\overline{\mathtt{A}} \sqcap \exists \mathtt{r}.\top$ becomes $\bot$-equivalent if $\mathtt{A}$ is replaced by $\bot$; the same holds with cardinality restrictions in place of "$\exists$". Consequently, axioms such as $\mathtt{A}^{\bot} \equiv \mathtt{B} \sqcap \forall \mathtt{r}.\mathtt{C}^{\bot} \sqcap \forall \mathtt{s}.\{a\} \sqcap =3\,\mathtt{r}.\top$, (taken from Koala) are $\emptyset$-local but not $\bot$-local.

We found this pattern in 8 ontologies. Only in miniT and Tambis, it affects a large proportion of bottom- and nested-modules, with additional axioms "pulled in". Still, the size of the differences is modest, as argued above. Some of the remaining 6 ontologies contain different kinds of complex classes that cause differences in top-locality of single axioms.

Culprits of Type *d* are axioms where a class (or property) name from the left-hand side occurs on the right-hand side together with a top-equivalent property (or class), causing differences in top-modules. The simplest such axiom is $\mathtt{A} \sqsubseteq \exists \overline{\mathtt{r}}.\mathtt{A}$, which is Δ-local because replacing r with $\top$ makes it a tautology. The axiom is only $\top$-local if Σ contains neither $\mathtt{r}$ nor $\mathtt{A}$. We have found further, more complex, examples in Adverse Event Reporting Ontology and Galen; see [5].

We have observed culprits of type d in 17 ontologies, see the detailed overview in [5]. Only in 3 cases (NBO, PhaRe, and TOK) are modules affected.

Galen contains 121 culprit-d axioms, but they only affect locality of single axioms. The time differences for Galen are remarkable:checking all axioms for Δ-locality takes up to 70 times longer than checking them for $\top$-locality.

Module Sizes. The selection of the signatures for the experiment was designed to allow for the analysis of two, complementary, kinds of modules: 1) genuine modules, which constitute a base of all modules, extracted from generally small axiom signatures; 2) a statistically significant amount of random modules, obtained from random, unbiased signatures which are likely to contain half the terms of the ontology. We argue in what follows that it is neither the case that genuine modules are so small to be almost irrelevant sets to investigate, nor that random modules are so big to leave no space for differences to be observed. We will focus on syntactic modules which contain the other kinds of modules.

During the experiment we have computed and analyzed a high number of genuine modules: more than 380K for the $\bot$-notion, more than 40K for the $\top$-notion, and more than 440K for the $\top\bot^*$-notion of locality. As we mentioned above, these modules tend to be quite small. However, they are not of irrelevant size: $\sim 8\%$ of the genuine $\bot$-modules, $\sim 11\%$ of the genuine $\top$-modules, and $\sim 5\%$ of the genuine $\top\bot^*$-modules contain more than 20% of the axioms of the corresponding ontology. So the low number of differences observed is not due to checking only against very small modules.

With a similar and complementary discussion, we argue that the modules obtained through random, "big" signatures do not necessarily contain almost all of the ontology: e.g., 39% of all random $\top\bot^*$-modules, and 28% of all random $\bot$-modules, contain less that 60% of the axioms of the corresponding ontology.

To sum up, the lack of differences between the modules is not due to too small or to too big sizes of the modules selected.

Discussion. All culprits hardly ever cause significant differences in modules. Only for PhaRe are differences between semantic and syntactic modules not negligible, but we were able to relativize them, see [5].

Table 1 may suggest that culprits occur only in expressive ontologies. However, patterns a, c, d can, in principle, already occur in simple terminologies in $\mathcal{EL}$ and $\mathcal{ALC}$, respectively. Evidently, type-a culprits can easily be filtered out in a preprocessing step. For types c and d, there is no hope for an exhaustive extension to locality because they can (and do) occur in arbitrarily complex shapes and contexts. For this reason, the identification of culprits can only be done "on demand", i.e., by observing the differences in the modules of given ontologies.

Patterns of type b rely on nominals or datatypes – but they are *repairable* by a straightforward extension to the definition of syntactic locality: one can extend the locality definition to distinguish $\bot$- and $\top$-*distinct* classes, by adding appropriate grammars to the definition of syntactic locality, and adding more cases of $\bot$- and $\top$-equivalent classes to the existing grammars. However, from the small numbers of differences observed, we doubt that such an extension of syntactic locality will have any significant effects in practice.

4.2 LBMs vs MEX Results

The results of the experimental comparison of syntactic/semantic LBMs and MEX modules are summarized in Table 3. They show that MEX modules smaller than the corresponding LBMs can be found in $\sim$27% of the preprocessed ontologies, for either random or axiom-based seed signatures. At the same time, unsurprisingly, syntactic and semantic LBMs do not differ at all for these simple $\mathcal{ELI}$ ontologies.

In experiments with random seed signatures, it can be seen that for those ontologies where there *are* differences (most notably, Galen), they occur in many tests. Thus, the difference appears to be caused by features of the ontology, not some particular seed signatures. Also, the difference sometimes comes out large in certain tests, also for genuine modules. For example, for the signature of the

Table 3. Differences between MEX and LBMs ($\top\bot^*$, $\mathit{\Delta}\emptyset^*$)

Experiment	#ontol. with diffs.	% tests with diffs.	avg size of diffs #axs	rel.
Random signatures	66	84%	0–26	0–13%
Axiom signatures	61	12%	0–13	0–80%

The results from the third column on are averaged over all ontologies with differences LBM–MEX in at least one module. For example, the last two columns show the average min and max absolute (resp. relative) difference between LBMs and MEX modules.

following axiom in Galen, both $\mathit{\Delta}\emptyset^*$-mod and $\top\bot^*$-mod contain 127 axioms while the MEX-module only contains the axiom itself:[8] `RICF` $\equiv$ `ICF` $\sqcap \exists$`ISFO.RSH`.

We analyzed whether the differences observed correlate with the size of the original ontology, its expressivity or the extent of the modification done in the $\mathcal{ELI}$-fication. There is no correlation with size but, as is to be expected, with the other two features, which are closely connected to each other. Table 4 illustrates the observations by dividing the 239 ontologies tested into four groups. The ontologies in Group 1 are in a format MEX can handle, so they have not been modified. The others required more or less heavy modifications (Groups 2–4). Differences between MEX and LBMs as described above occur only for ontologies that required heavy modifications (Group 4).

Table 4. Overview of MEX experiment

Group		#axioms removed	#ontologies	ontology size (avg.)
1	unchanged ontologies *no* diff. $\mathit{\Delta}\emptyset^* \setminus$ MEX	0	33 (14%)	19–16,066 (2,176)
2	little-changed ontologies *no* diff. $\mathit{\Delta}\emptyset^* \setminus$ MEX	1–28	36 (15%)	13– 6,587 (466)
3	largely-changed ontologies *no* diff. $\mathit{\Delta}\emptyset^* \setminus$ MEX	31–7,836 (avg. 884)	104 (44%)	51–13,153 (2,373)
4	largely-changed ontologies *with* diff. $\mathit{\Delta}\emptyset^* \setminus$ MEX	30–12,185 (avg. 1,001)	66 (27%)	42–12,344 (1,843)

As expected, the expressivity among Groups 1 and 2 is generally low: only 21 ontologies in Group 2 use expressivity above $\mathcal{ALE}$ (up to $\mathcal{SHIF}(\mathcal{D})$, which is an outlier). However, the size of some ontologies in Group 1 is already considerable: 22 out of 33 have $>$ 100 axioms; 10 have $>$ 1,000 axioms. In contrast, the ontologies in Group 4 have almost always high expressivity, for example 27 out of 66 contain nominals.

[8] The acronyms denote RightIneffectiveCardiacFunction, IneffectiveCardiacFunction, isSpecificFunctionOf, RightSideOfHeart.

Despite the correlation between the impact of the $\mathcal{ELI}$-fication and the differences observed between MEX- and $\Delta\emptyset^*$-modules, we cannot claim that there is a causation between the two events. Indeed, we have investigated the reasons for the differences observed between the two kinds of modules, and we have noticed that in all the cases the culprit is the proliferation of equivalence axioms. For example $\mathtt{A} \equiv \mathtt{B}$ will end up in the $\Delta\emptyset^*$-mod for any seed signature containing either A or B. It is, however, an mCE of $\emptyset$ w.r.t. to either $\{\mathtt{A}\}$ or $\{\mathtt{B}\}$.

The experimental results in view of this insight are summarized as follows:

Random-modules experiment: the 66 ontologies where differences between random MEX- and $\Delta\emptyset^*$-modules were observed, coincide exactly with those where equivalences occur in the $\mathcal{ELI}$-TBox.

Genuine-modules experiment: all 61 ontologies where differences between genuine MEX- and $\Delta\emptyset^*$-modules were observed contain equivalence axioms.

We conjecture that the low expressivity of the $\mathcal{ELI}$-language reduce the possibility of MEX- and $\Delta\emptyset^*$-modules to differ only to the presence of equivalences. In addiction to the empirical evidence for such a claim, we plan to investigate further this aspect in future work.

5 Conclusion and Outlook

Summary. We obtain three main observations from our experiments. (1) In general, there is no or little difference between semantic and syntactic locality. Hence, the computationally cheaper syntactic locality is a good approximation of semantic locality. (2) In most cases, there is no or little difference between LBMs and MEX modules. (3) Though in principle hard to compute, semantic LBMs can be extracted rather fast in practice. Still, their extraction often takes considerably longer than that of syntactic LBMs. We cannot make any statement about MEX module extraction times because we use the original MEX implementation, which combines loading and module extraction. Due to results (1) and (2), hardly any benefit can be expected from preferring potentially smaller modules (MEX or semantic LBMs) to cheaper syntactic LBMs. For the ontologies Galen and People, which are "renowned" for having disproportionately large modules, syntactic and semantic LBMs do not differ. Only for Galen are MEX modules considerably smaller than LBMs.

Not only does our study evaluate how good the cheap syntactic locality approximates semantic locality and model conservativity, it also required us to provide the first implementation for extracting modules based on semantic locality. Furthermore, we have been able to fix bugs in the existing implementation of syntactic modularity. A complete report of bugfixes is beyond the scope of the paper; as an example, early runs of the experiment led us to correcting the treatment of reflexivity axioms by the locality checker in the OWL API.

Future Work. Two issues are interesting for future work: (1) Sampling seed signatures so that all sizes of signatures are equally likely to be sampled; (2) Comparing LBMs to other types of conservativity-based modules.

As for (1), the current sampling causes small and large signatures to be underrepresented. One might argue that, for big ontologies, the typical module extraction scenario does not require large seed signatures – but it does sometimes require relatively small seed signatures, for example, when a module is extracted to efficiently answer a certain entailment query of typically small size. We therefore plan to conduct a similar experiment using other sampling methods. Concerning (2), one could include, for example, the technique based on reduction to QBF for the OWL 2 QL profile [16] when an off-the-shelf implementation becomes available.

References

1. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P.F. (eds.): The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press (2003)
2. Croarkin, C., Tobias, P. (eds.): NIST/SEMATECH e-Handbook of Statistical Methods. NIST/SEMATECH (2012), `http://www.itl.nist.gov/div898/handbook`
3. Cuenca Grau, B., Horrocks, I., Kazakov, Y., Sattler, U.: Modular reuse of ontologies: Theory and practice. J. of Artif. Intell. Research 31(1), 273–318 (2008)
4. Cuenca Grau, B., Parsia, B., Sirin, E., Kalyanpur, A.: Modularity and Web ontologies. In: Proc. of KR 2006. AAAI Press/The MIT Press (2006)
5. Del Vescovo, C., Klinov, P., Parsia, B., Sattler, U., Schneider, T., Tsarkov, D.: Empirical study of logic-based modules: Cheap is cheerful. Technical report (2013), `https://sites.google.com/site/cheapischeerful/`
6. Del Vescovo, C., Parsia, B., Sattler, U., Schneider, T.: The modular structure of an ontology: an empirical study. In: Proc. of DL 2010, vol. 573. ceur-ws.org (2010)
7. Del Vescovo, C., Parsia, B., Sattler, U., Schneider, T.: The modular structure of an ontology: Atomic decomposition. In: Proc. of IJCAI 2011, pp. 2232–2237 (2011)
8. Del Vescovo, C., Parsia, B., Sattler, U., Schneider, T.: The modular structure of an ontology: Atomic decomposition and module count. In: Proc. of WoMO 2011. FAIA, vol. 230, pp. 25–39 (2011)
9. Garson, J.: Modularity and relevant logic. Notre Dame J. of Formal Logic 30(2), 207–223 (1989)
10. Ghilardi, S., Lutz, C., Wolter, F.: Did I damage my ontology? A case for conservative extensions in Description Logics. In: Proc. of KR 2006, pp. 187–197. AAAI Press/The MIT Press (2006)
11. Horridge, M., Parsia, B., Sattler, U.: Laconic and precise justifications in OWL. In: Sheth, A.P., Staab, S., Dean, M., Paolucci, M., Maynard, D., Finin, T., Thirunarayan, K. (eds.) ISWC 2008. LNCS, vol. 5318, pp. 323–338. Springer, Heidelberg (2008)
12. Horridge, M., Parsia, B., Sattler, U.: The state of bio-medical ontologies. In: Proc. of ISMB 2011 (2011)
13. Horrocks, I., Kutz, O., Sattler, U.: The even more irresistible $\mathcal{SROIQ}$. In: Proc. of KR 2006, pp. 57–67 (2006)
14. Konev, B., Lutz, C., Walther, D., Wolter, F.: Semantic modularity and module extraction in description logics. In: Proc. of ECAI 2008, pp. 55–59 (2008)
15. Kontchakov, R., Pulina, L., Sattler, U., Schneider, T., Selmer, P., Wolter, F., Zakharyaschev, M.: Minimal module extraction from DL-Lite ontologies using QBF solvers. In: Proc. of IJCAI 2009, pp. 836–841 (2009)

16. Kontchakov, R., Wolter, F., Zakharyaschev, M.: Logic-based ontology comparison and module extraction, with an application to DL-Lite. Artificial Intelligence 174(15), 1093–1141 (2010)
17. Lutz, C., Walther, D., Wolter, F.: Conservative extensions in expressive Description Logics. In: Proc. of IJCAI 2007, pp. 453–458 (2007)
18. Lutz, C., Wolter, F.: Deciding inseparability and conservative extensions in the description logic $\mathcal{EL}$. J. of Symbolic Computation 45(2), 194–228 (2010)
19. Sattler, U., Schneider, T., Zakharyaschev, M.: Which kind of module should I extract? In: Proc. of DL 2009, vol. 477. ceur-ws.org (2009)
20. Smithson, M.: Confidence Intervals. Quantitative Applications in the Social Sciences. Sage Publications (2003)

The Logic of Extensional RDFS

Enrico Franconi[1], Claudio Gutierrez[2], Alessandro Mosca[1], Giuseppe Pirrò[1], and Riccardo Rosati[3]

[1] KRDB, Free University of Bozen-Bolzano, Bolzano, Italy
[2] Department of Computer Science, University of Chile, Santiago, Chile
[3] University of Rome La Sapienza, Rome, Italy

Abstract. The normative version of RDF Schema (RDFS) gives non-standard (intensional) interpretations to some standard notions such as classes and properties, thus departing from standard set-based semantics. In this paper we develop a standard set-based (extensional) semantics for the RDFS vocabulary while preserving the simplicity and computational complexity of deduction of the intensional version. This result can positively impact current implementations, as reasoning in RDFS can be implemented following common set-based intuitions and be compatible with OWL extensions.

1 Introduction

The Resource Description Framework (RDF) [9] is the standard data model for publishing and interlinking data on the Web. Its associated vocabulary RDF Schema (RDFS) (classes, properties, hierarchies) gives non-standard (intensional) interpretations to some standard set theoretical notions such as classes and properties. This brings some difficulties to the reasoning systems based on classical first-order logic (FOL). RDF enables the making of *statements* about (Web) resources in the form of triples including a *subject*, a *predicate* and an *object* expressed in manifold vocabularies. Efforts like the Linked Open Data project [8] give a glimpse of the magnitude of RDF data today available.

In many application scenarios, there is the need to have on top of RDF data a language to structure knowledge domains. To cope with this aspect, the Web Consortium developed standard vocabularies such as RDF Schema (RDFS) and OWL. RDFS was designed with a minimalist philosophy and it includes essentially the machinery for expressing subclass, subproperty, type and such. On the other hand, OWL is a more expressive language that includes a much richer set of features.

From a standardization point of view the current normative RDFS has two weaknesses. First, the interpretations of basic notions such as subclass and subproperty do not have the usual set-based meaning. For example, in in Fig. 1, even though `:birthCity` is a subproperty `:birthPlace`, one cannot derive the fact that the range of the property `:birthCity` must be `:Place`. Second, the normative semantics of RDFS and OWL differ for some of their common vocabularies. RDFS, for historical reasons, follows an *intensional* semantics while OWL

H. Alani et al. (Eds.): ISWC 2013, Part I, LNCS 8218, pp. 101–116, 2013.

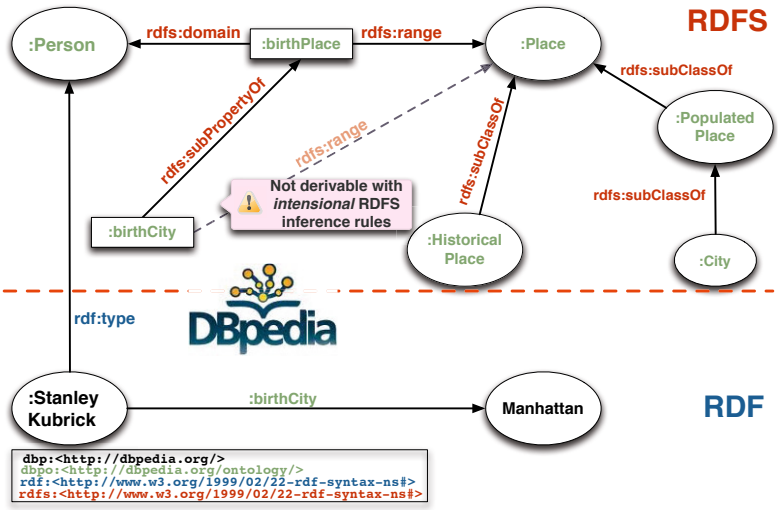


Fig. 1. An RDFS graph taken from `dbpedia.org` showing compatibility problems between OWL and RDFS. The dotted arrow is valid in OWL while not in RDFS.

adopts a standard *extensional* set-based semantics. This intensional semantics of RDFS brings compatibility problems with OWL. In the example considered shown in Fig. 1, the dotted `rdfs:range` property is a valid set-based deduction, thus valid in OWL, while *not* derivable in RDFS.

The designers of RDFS were aware of this problem, and added in a "non-normative" status the standard set-based semantics and some sound inference rules for it. This so-called "extensional" version of RDFS corresponds exactly to the standard set-based interpretation of the vocabulary (and thus is fully compatible with OWL). The rationale for keeping a weaker (intensional) semantics for RDFS was efficiency: "In some ways the extensional versions provide a simpler semantics, but they require more complex inference rules. The 'intensional' semantics [...] provides for most common uses of subclass and subproperty assertions, and allows for simpler implementations of a complete set of RDFS entailment rules." (W3C RDFS Semantics Spec., [7]). According to this specification, RDFS inference engines develop following the intensional semantics.

Thus, two relevant problems regarding the natural extensional RDFS semantics have prevented its usage: *i)* What is the complexity overload associated to the extensional semantics for RDFS?; *ii)* Can normative RDFS inference engines (based on the computation of a completion in a forward-chaining manner) be easily extended to support extensional RDFS, and at which cost?

Contributions. This paper answer both question in the positive. First, we provide a simple sound and complete proof system for the extensional semantics of RDFS. Second, we show that a meaningful completion of the graph computed by using the rules in a forward-chaining manner can still be computed in

polynomial case (as for intensional RDFS) thus spurring on current system that use completion. These two results can be seen as founding the ground for the developing of the extensional semantics for the RDFS vocabulary while preserving the simplicity and computational complexity of deduction of the intensional case.

Our results can be considered as an extension of intensional RDFS. Our results address not only an interesting theoretical open problem, but could impact on current implementations (for the most part based on the normative intensional semantics) in a positive sense. Indeed, we show that reasoning in RDFS can follow common set-based intuitions and be compatible with OWL extensions. Moreover, we show that the rule system that we present is easily embeddable in existing libraries such as Jena.

2 Preliminaries

The Resource Description Framework (RDF) [9] and RDF Schema (RDFS) are the W3C's standard data model for the publishing and interlinking of data on the Web. In RDF only simple statements about resources can be expressed via triples: a resource may be an instance of another resource (representing a class typing the instance) and/or a property of another resource. RDFS augments RDF with some minimal *vocabulary*, allowing to express hierarchies of classes and properties and to restrict the domain and range of properties. As an example of an RDFS graph, see Fig. 1. In what follow we will give a simple presentation abstracted from implementation (e.g., namespace) details.

Let $\mathcal{U}$, $\mathcal{L}$, $\mathcal{B}$ three pairwise disjoint sets representing the set of URIs, literals and blank nodes, respectively. For simplicity, we denote unions of these sets by simply concatenating their names.

Definition 1 (RDF triple, graph). *An RDF triple t is a tuple of the form $(s, p, o) \in (\mathcal{UB}) \times \mathcal{U} \times (\mathcal{UBL})$, where s, p, o are called subject, predicate and object, respectively. A triple is ground if it does not contain blank nodes. A (ground) RDF graph $\mathcal{G}$ is a set of RDF (ground) triples. The vocabulary of $\mathcal{G}$, denoted* $\mathrm{voc}(\mathcal{G})$*, is the set of elements in $\mathcal{UBL}$ that occurs in its triples.*

The ρdf Fragment

In this work we will concentrate on a simple and small fragment of RDFS, which includes only the special RDFS vocabulary `type`, `property`, `subClass`, `subProperty`, `domain` and `range`. This fragment is called **ρdf** and was introduced first in [11]. It has been shown to capture the essential semantics of the full fragment, while avoiding to deal with minor idiosyncrasies. In the following, we will denote its vocabulary as $\mathcal{V}_{\rho\mathrm{df}} = \{\mathtt{sc}, \mathtt{sp}, \mathtt{dom}, \mathtt{range}, \mathtt{type}\}$. As it has been shown in [11], ρdf is self-contained as it does not rely on the RDFS vocabulary beyond this subset. ρdf is endowed with a set of inference rules that preserves the original RDFS semantics restricted to this vocabulary [10].

The Intensional (Normative) Semantics

The normative semantics of RDFS [7] is built upon the standard logic notions of model, interpretation and entailment. In the following we rephrase the normative

model theory of RDFS using first-order logic (FOL) in the spirit of [4]. The signature of the language includes a ternary predicate T –to represent RDF triples – and two unary predicates C and P that will represent the membership of individuals to "`rdfs:Class`" and "`rdf:Property`", respectively. It can be proved that, given a ρdf graph $\{(s_1, p_1, o_1), \cdots, (s_n, p_n, o_n)\}$, its models according the the normative RDFS model theory in the W3C specification [4] are the same as the models of the FOL formula $\exists \mathbf{b}\, T(s_1, p_1, o_1) \wedge \cdots \wedge T(s_n, p_n, o_n)$, where $\mathbf{b}$ is the set of blank node symbols appearing in the graph, under the FOL theory specified by the axioms listed below.

The basic axioms primitively define `subClass`, `subProperty`, `domain`, `range` in terms of `type` in the obvious way –as in set theory[1]:

$$\forall a, b\ (a, \mathtt{sc}, b) \longrightarrow C(a) \wedge C(b) \wedge \forall x\ (x, \mathtt{type}, a) \rightarrow (x, \mathtt{type}, b) \quad (1)$$

$$\forall a, b\ (a, \mathtt{sp}, b) \longrightarrow P(a) \wedge P(b) \wedge \forall x, y\ (x, a, y) \rightarrow (x, b, y) \quad (2)$$

$$\forall a, c\ (a, \mathtt{dom}, c) \longrightarrow \forall x, y\ (x, a, y) \rightarrow (x, \mathtt{type}, c) \quad (3)$$

$$\forall a, d\ (a, \mathtt{range}, d) \longrightarrow \forall x, y\ (x, a, y) \rightarrow (y, \mathtt{type}, d) \quad (4)$$

To cope with reflexivity and transitivity of the subclass and subproperty relations we have also the following axioms:

$$\forall a, b, c\ (a, \mathtt{sc}, b) \wedge (b, \mathtt{sc}, c) \longrightarrow (a, \mathtt{sc}, c) \quad (5)$$

$$\forall a\ C(a) \longrightarrow (a, \mathtt{sc}, a) \quad (6)$$

$$\forall a, b, c\ (a, \mathtt{sp}, b) \wedge (b, \mathtt{sp}, c) \longrightarrow (a, \mathtt{sp}, c) \quad (7)$$

$$\forall a P(a) \longrightarrow (a, \mathtt{sp}, a) \quad (8)$$

The following typing axioms are also needed in normative RDFS:

$$\forall a, b\ (a, \mathtt{dom}, b) \longrightarrow P(a) \wedge C(b) \quad (9)$$

$$\forall a, b\ (a, \mathtt{range}, b) \longrightarrow P(a) \wedge C(b) \quad (10)$$

$$\forall a, b\ (a, \mathtt{type}, b) \longrightarrow C(b) \quad (11)$$

$$\forall a, b, c\ (a, b, c) \longrightarrow P(b) \quad (12)$$

$$P(\mathtt{sc}) \wedge P(\mathtt{sp}) \wedge P(\mathtt{dom}) \wedge P(\mathtt{range}) \wedge P(\mathtt{type}) \quad (13)$$

The above axioms define the semantics for the `subClass`, `subProperty`, `domain` and `range` predicates.

It is important to observe that `rdfs:subClass`, `rdfs:subProperty`, `rdfs:domain`, `rdfs:range` are defined only by means of *necessary* properties according to the above axioms: the semantics of normative RDFS is a quite weak one, since the RDFS vocabulary does not express fully the corresponding relations in set theory. As a matter of facts, given the RDFS graph from Fig. 1, according the normative RDFS semantics the statement (`:birthCity`, `rdfs:range`, `:Place`) is not entailed. Such an entailment is expected since people do read the properties in the RDFS vocabulary as the corresponding set-based relations – just like in

[1] Note that for simplicity we may omit the T symbol in FOL formulas.

OWL. The normative RDFS semantics is called *intensional*, since it is unable to define sets in terms of their elements.

The Extensional (Non-normative) Semantics
The W3C specification [7] introduces in a "non-normative"status an *extensional* version of RDFS, in which `subClass`, `subProperty`, `domain`, `range` are defined precisely as having the usual set theoretical meaning. This is achieved by adding to the previous definition of the RDFS semantics the missing implication (left-direction arrows) in axioms (1) to (4), thus getting axioms (14) to (17). Thus, axioms (1) to (17) define the semantics of the non-normative extensional RDFS restricted to the ρdf vocabulary. Note that axioms (1) to (8) are redundant, since they can be derived from axioms (9) to (17). From now on we will refer to the non-normative version of RDFS restricted to the ρdf vocabulary as $\boldsymbol{\rho}$**df+**.

$$\forall a, b\ (a, \mathtt{sc}, b) \longleftrightarrow C(a) \wedge C(b) \wedge \forall x\ (x, \mathtt{type}, a) \rightarrow (x, \mathtt{type}, b) \tag{14}$$

$$\forall a, b\ (a, \mathtt{sp}, b) \longleftrightarrow P(a) \wedge P(b) \wedge \forall x, y\ (x, a, y) \rightarrow (x, b, y) \tag{15}$$

$$\forall a, c\ (a, \mathtt{dom}, c) \longleftrightarrow \forall x, y\ (x, a, y) \rightarrow (x, \mathtt{type}, c) \tag{16}$$

$$\forall a, d\ (a, \mathtt{range}, d) \longleftrightarrow \forall x, y\ (x, a, y) \rightarrow (y, \mathtt{type}, d) \tag{17}$$

This (extensional) semantics – which follows exactly the obvious extensional definitions of the corresponding set-based operators – has been disregarded by the W3C working group because of some computational problems that were conjectured during the definition of the specification. In the non normative section of the W3C specification only a set of *incomplete* inference rules for extensional RDFS is provided.

As for the relations with other KR formalisms, and with the family of description logics in particular, notice that it is easy to see that ρdf+ without typing *exactly* corresponds to the DL-Lite$^{\mathcal{H}}_{\{\text{core,pos,safe}\}}$, namely the well known DL-Lite$^{\mathcal{H}}_{\{\text{core}\}}$ description logic [2,1] without negation and unqualified existential restrictions on the right-hand side of the inclusion axioms. Obviously, DL-Lite$^{\mathcal{H}}_{\{\text{core,pos,safe}\}}$ *includes* the normative RDFS. It is easy to see that the usual unqualified number restrictions of DL-Lite$_{core}$, once on the left-hand side of the inclusion axioms, can be used to encode the `rdfs:domain` and `rdfs:range` statements, while `rdfs:subClass` and `rdfs:subProperty` are nothing but usual *DL* concept and role inclusion axioms, respectively.

Although the semantics of RDFS dates back to 2004 and despite the large amount of research around it, there were still some important open problems concerning extensional RDFS: i) whether a sound and complete system of inference rules existed; ii) whether a polynomial algorithm for computing the completion according to these extensional rules existed; iii) whether the problem of entailment checking, crucial for query answering, can still be done in the same complexity bound as for intensional RDFS. In this paper we tackle these three problems and provide positive answers to each of them.

3 Reasoning with ρdf+: A Forward-Chaining System

This section presents a set of sound and complete inference rules for ρdf+ that captures the extensional semantics of RDFS. Our findings complement the set of rules in the ρdf fragment with additional rules derived from the analysis of axioms (14)-(17). The complete set of rules is presented in Table 1. For example, the missing deduction in Fig. 1 can be done now with rule 4(b) with the instantiations $A =$`birthCity`, $B =$`birthPlace` and $C =$`Place`.

We will need some definitions for the discussion that follows. We follow the notations of [11].

Definition 2 (Instantiations and maps)

1. *An* instantiation *of a rule is a uniform replacement of the meta variables occurring in the triples of the rule with elements in $\mathcal{UBL}$, such that all the triples obtained after the replacement are well-formed RDF triples.*
2. *A map is a function $\mu : \mathcal{UBL} \to \mathcal{UBL}$ preserving URIs and literals i.e., $\mu(u) = u$ for all $u \in \mathcal{UL}$. Given a graph $\mathcal{G}$ we define $\mu(\mathcal{G}) = \{(\mu(s), \mu(p), \mu(o)) : (s, p, o) \in \mathcal{G}\}$. By abusing notation, we speak of a map μ from a graph $\mathcal{G}_1$ to a graph $\mathcal{G}_2$ and write $\mu : \mathcal{G}_1 \to \mathcal{G}_2$ if μ is such that $\mu(\mathcal{G}_1)$ is a subgraph of $\mathcal{G}_2$.*

Definition 3 (Proof). *Let $\mathcal{G}$ and $\mathcal{H}$ be graphs. We say that $\mathcal{G} \vdash_{\rho df+} \mathcal{H}$ iff there exists a sequence of graphs $P_1, P_2, \ldots, P_k$, with $P_1 = \mathcal{G}$ and $P_k = \mathcal{H}$, and for each j $(2 \leq j \leq k)$ one of the following cases hold:*

- *there exists a map $\mu : P_j \to P_{j-1}$ (rule 8),*
- *there is an instantiation $\frac{R}{R'}$ of one of the rules (1)–(7) in Table 1 such that $R \subseteq P_{j-1}$ and $P_j = P_{j-1} \cup R'$.*

The sequence of rules used at each step (plus its instantiation or map), is called a proof *of $\mathcal{H}$ from $\mathcal{G}$.*

The ρdf+ system of rules extends the ρdf system [11] by the rules 3(b), 3(c), 4(b), 4(c) and (7). The following theorem states the soundness and completeness of $\vdash_{\rho df+}$.

Theorem 1 (Soundness and completeness). *Let $\models_{\rho df+}$ denote the entailment relation for the extensional ρdf+ semantics obtained from the axioms (1)-(17). Then, the proof system $\vdash_{\rho df+}$ (rules in Table 1) is sound and complete for this extensional semantics; that is, for $\mathcal{G}$ and $\mathcal{H}$ graphs in ρdf+, then $\mathcal{G} \vdash_{\rho df+} \mathcal{H}$ iff $\mathcal{G} \models_{\rho df+} \mathcal{H}$.*

Proof. The proof is available in the Appendix. □

Although the natural consequence of Theorem 1 would be that of dropping the intensional (weaker) semantic conditions in the normative semantics and replacing them with the extensional (stronger), it is still necessary to investigate whether ρdf+ brings in some source of complexity when applied to the

Table 1. The $\vdash_{\rho\text{df}+}$ rule system for ρdf+. Capital letters A, B, C, X, and Y, stand for meta-variables to be replaced by actual terms in $\mathcal{UBL}$.

1. **Subclass**:
   (a) $\frac{(A,\texttt{sc},B)\ \ (X,\texttt{type},A)}{(X,\texttt{type},B)}$ (b) $\frac{(A,\texttt{sc},B)\ \ (B,\texttt{sc},C)}{(A,\texttt{sc},C)}$
2. **Subproperty**:
   (a) $\frac{(A,\texttt{sp},B)\ \ (X,A,Y)}{(X,B,Y)}$ (b) $\frac{(A,\texttt{sp},B)\ \ (B,\texttt{sp},C)}{(A,\texttt{sp},C)}$
3. **Domain:**
   (a) $\frac{(A,\texttt{dom},B)\ \ (X,A,Y)}{(X,\texttt{type},B)}$ (b) $\frac{(A,\texttt{sp},B)\ \ (B,\texttt{dom},C)}{(A,\texttt{dom},C)}$ (c) $\frac{(A,\texttt{dom},B)\ \ (B,\texttt{sc},C)}{(A,\texttt{dom},C)}$
4. **Range:**
   (a) $\frac{(A,\texttt{range},B)\ \ (X,A,Y)}{(Y,\texttt{type},B)}$ (b) $\frac{(A,\texttt{sp},B)\ \ (B,\texttt{range},C)}{(A,\texttt{range},C)}$ (c) $\frac{(A,\texttt{range},B)\ \ (B,\texttt{sc},C)}{(A,\texttt{range},C)}$
5. **Subclass Reflexivity:**
   (a) $\frac{(A,\texttt{sc},B)}{(A,\texttt{sc},A)\ \ (B,\texttt{sc},B)}$ (b) $\frac{(X,p,A)}{(A,\texttt{sc},A)}$ for $p \in \{\texttt{dom},\texttt{range},\texttt{type}\}$
6. **Subproperty Reflexivity:**
   (a) $\frac{(X,A,Y)}{(A,\texttt{sp},A)}$ (c) $\frac{}{(p,\texttt{sp},p)}$ for $p \in \rho\text{df}$
   (b) $\frac{(A,\texttt{sp},B)}{(A,\texttt{sp},A)\ \ (B,\texttt{sp},B)}$ (d) $\frac{(A,p,X)}{(A,\texttt{sp},A)}$ for $p \in \{\texttt{dom},\texttt{range}\}$
7. **Extensional:**
   $\frac{(\texttt{type},\texttt{sp},A)\ \ (A,\texttt{dom},B)\ \ (X,\texttt{sc},X)}{(X,\texttt{sc},B)}$
8. **Simple:**
   $\frac{\mathcal{G}}{\mathcal{G}'}$ for a map $\mu : \mathcal{G}' \to \mathcal{G}$

following important reasoning tasks: i) computation of the closure; ii) checking of entailment, crucial for query answering.

Computational Properties of ρdf+

The *deductive closure* of a graph $\mathcal{G}$ is the graph obtained by adding to $\mathcal{G}$ all triples that are derivable from $\mathcal{G}$. It can be computed by applying systematically and recursively the inference rules in Table 1 to all the triples of $\mathcal{G}$. The deductive closure of a ρdf+ graph is in principle infinite, due to the rule 8, which possibly introduces new blank nodes. In order to get a finite but still useful *completion* of the graph we can consider the closure of $\mathcal{G}$ over the same vocabulary of $\mathcal{G}$, that is, by adding only triples derivable from $\mathcal{G}$ which have elements in $\text{voc}(\mathcal{G}) \cup \mathcal{V}_{\rho\text{df}}$. We will denote this restricted closure by $cl_g(\mathcal{G})$ be the *ground closure* (or *completion*)

of a graph $\mathcal{G}$ as the closure via the $\vdash_{\rho\text{df}+}$ ground rule system (rules (1)-(7) in Table 1).

By observing that the number of existing triples with vocabulary in $\text{voc}(\mathcal{G}) \cup \mathcal{V}_{\rho\text{df}}$ is of the order $O(|\mathcal{G}|^3)$, and that all new triples in the closure of $\mathcal{G}$ will be obtained by a successive applications of the rules of the proof system, we obtain the following result:

Proposition 1 (Closure complexity). *The size of the ground closure of a ρdf graph $cl_g(\mathcal{G})$ is at most $O(|\mathcal{G}|^3)$ and it can be computed in polynomial time.*

We will now present a result which states how ρdf+ entailment can be constructively reduced to computing (possibly offline) and materialising the finite polynomial completion of the data graph and then by querying the completion with a standard RDF *simple entailment* query engine. Note that this is the very same procedure which is used in real systems for the standard normative RDFS entailment – of course with the reduced set of normative RDFS inference rules.

Proposition 2 (Entailment for ρdf+). *Consider two RDFS graphs $\mathcal{G}$ (data) and $\mathcal{H}$ (pattern). Then $\mathcal{G} \models_{\rho df+} \mathcal{H}$ iff $cl_g(\mathcal{G}) \models_{\text{RDF}_{\text{simple}}} \mathcal{H}$.*

Proof. By the completeness theorem, $\mathcal{G} \vdash_{\rho\text{df}+} \mathcal{H}$, which by definition of the closure is equivalent to $cl_g(\mathcal{G}) \vdash_{\rho\text{df}+} \mathcal{H}$, which means that $\mathcal{H}$ is in the completion $cl_g(\mathcal{G})$, unless there is an application of rule 8. In this case, $\mathcal{H}$ is got by using the RDF simple entailment in the entailment checking –because of the homomorphism checking. □

It can be easily seen that the combined complexity of entailment (in the size of both graphs) is exactly the same as for normative RDFS and the ρdf system, which is polynomial if $\mathcal{H}$ is a ground graph, and NP-hard otherwise [11]. On the other hand, the data complexity of entailment (that is, only in the size of the data graph $\mathcal{G}$) is polynomial [4].

Materializing all data by computing the completion may cause a waste of space if most of it is never really used. Deciding whether applying materialization or checking entailment on the fly with a specific algorithm depends on different factors such as: i) size of the graph: some graphs may not fit in the main memory and then the completion cannot be avoided; ii) updates: removing a triple from the graph, causes implicit data to still exist if no special care is taken to remove it. Hence, materialization vs. on the fly checking is a trade-off between the better performance of updates, or better performance of look-ups. For this purpose we have studied a refutation proof system provably sound and complete for ρdf+ based on tableaux calculus, which in addition to ρdf+ deals also with negative atoms in the data graph. Such a system, which we do not present here, is used to check entailment on the fly whenever it is not convenient to materialise the completion (see [5] for further details).

4 Reasoning with Extensional RDFS in Practice

The aim of this section is to illustrate with simple examples the practical impact of extensional RDFS reasoning. We discuss how the $\vdash_{\rho\text{df}+}$ system of rules can

be embedded into the Apache Jena library and the impact that it has on the computation of the completion of an RDFS graph.

The Jena Inference Engine
Jena is a comprehensive Semantic Web library providing a set of features for data management and reasoning in OWL and RDFS. The library features four predefined reasoning engines: i) *transitive reasoner*, which just considers transitive and reflexive properties of RDFS `sc` and `sp`; ii) a configurable *RDFS rule reasoner*; iii) a configurable *OWL reasoner*; iv) *a custom reasoner*. This latter reasoner enables to provide a custom set of inference rules; it supports three reasoning strategies: i) one implementing the *RETE algorithm*; ii) a *forward reasoner*; iii) a *backward reasoner*.

The availability of the custom reasoner is at the core of the integration of the *ground ρdf+ rule system*; we have not implemented rule 7, since we assume that data graphs do not redefine `rdf:type`, that is, they do not have it in subject nor object position. As an example the rule 3 (c) in Table 1 is specified in Jena as: `[3c: (?a dom ?b), (?b sc ?c)->(?a dom ?c)]`. The specification follows the pattern `[label: Ant ->Cons]` where `label` is a name assigned to the rule, `Ant` is the antecedent and `Cons` the consequent. It is also worth mentioning that the reasoner can be configured to log derivations so that each triple obtained after the reasoning task has associated an "explanation", that is, the reasoning steps (in terms of rules triggered) that led to the triple. The reader can consult the Web page `https://jena.apache.org/documentation/inference` for further details.

Comparing Inferences at Schema Level
We investigated the impact of ρdf+ on the completion of five existing ontologies. This experiment only considers triples at schema level; as discussed previously, we do not need to analyze derived `rdf:type` triples, since they would be the same as the `rdf:type` triples derived by a normative RDFS reasoner. Table 2 provides some information about the ontologies considered.

Table 2. Statistics about the ontologies considered

Ontology	#Classes	#Properties	#`dom`	#`range`	#`sc`	#`sp`
DBpedia	359	1775	1505	1553	369	-
FOAF	24	51	47	46	15	10
NEPOMUK	399	628	535	561	460	258
MusicOnto	70	97	97	97	68	25
VoxPopuli	140	66	61	78	140	-

The considered ontologies have different sizes; they range from small ontologies such as FOAF (Friend-of-a-Friend) or MusicOnto (Music Ontology) to relatively large ontologies like NEPOMUK and DBpedia. None of these (real-life) ontologies includes RDF triples redefining the RDFS vocabulary, that is, containing the

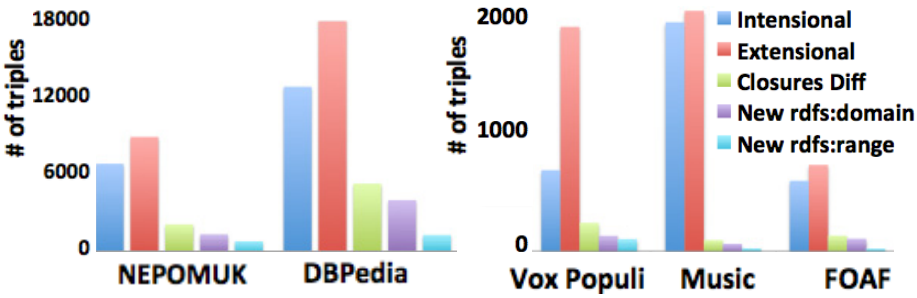


Fig. 2. Size of the completions

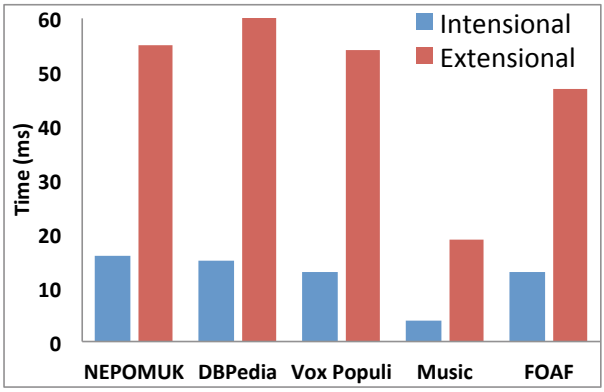


Fig. 3. Times for computing the completions

ρdf vocabulary in subject or object position. Fig. 2 shows some statistics about the completion of the ontologies by considering the ρdf (intensional RDFS) and ground ρdf+ (extensional RDFS) rule systems. The comparison between the completions in terms of number of triples is also shown . As it can be observed with ρdf+ we obtain a larger number of triples. This is due to the presence of the rules 3(b), 3(c), 4(b) and 4(c) in Table 1 that enable to derive new `rdfs:domain` and `rdfs:range` relations. The largest number was obtained when considering DBpedia (∼4000 `rdfs:domain` and ∼ 1200 `rdfs:range`). The extensional completion contains an increase of triples of the order of 30% for DBpedia and NEPOMUK, 60% for VoxPopuli, 20% for FOAF and 5% for MusicOnto. Fig. 3 reports the times (in ms) taken to compute the completion.

In the extensional case more time is needed because of the presence of additional inference rules. However, it can be observed that the time remains around 60ms with a large schema like DBpedia.

In order to give a hint on the kind of derivations enabled via ρdf+, Fig. 4 shows two examples from DBpedia. In Fig. 4 (a) it is shown the new `rdfs:range` for the property `:beltwayCity` obtained by applying rule 4 (c). Fig. 4 (b) shows

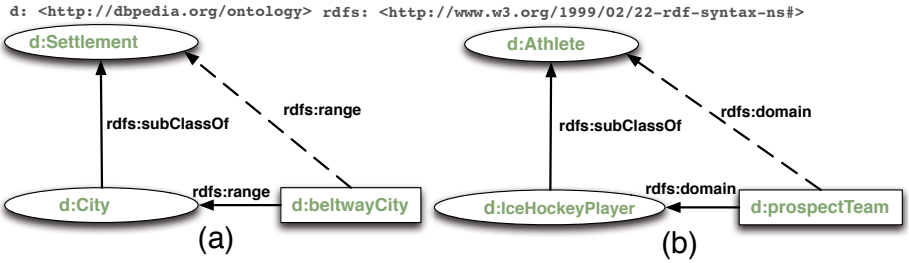


Fig. 4. Examples of new derivations with ρdf+

the derivation of a new `rdfs:domain` for the property `:prospectTeam` obtained via rule 3(c).

5 Related Work

There is a solid body of research on RDFS. A formalisation of RDF regarding databases issues was done by Gutierrez et al. [6]. Marin [10] and ter Horst [12] came up with counterexamples (see Fig. 1) which, though pointing to the incompleteness of the W3C RDF Semantics specification rules, showed an issue belonging to the intensional approach to RDFS. The merit of Marin was to overcome the issue keeping the original rules, and adding two additional ones, and proved that the new set of rules was sound and complete. ter Horst instead modified the rule system by allowing non-legal RDFS triples within the rule system by using blank nodes in the predicate position. The formalization of the semantics of RDF in FOL has been studied by de Brujin et al. [4]. Muñoz et al. [11] introduced the ρdf fragment; this paper also discusses the quadratic lower bound for the size of the completion of a graph $\mathcal{G}$ pointing out how such size is impractical from a database point of view. To cope with this issue, the authors introduce *minimal* RDFS, which imposes restrictions on the occurrence of the RDFS vocabulary (it can only occur in predicate position). The advantage of minimal RDFS is that there exists an efficient algorithm to check graph entailment in the case of ground graphs. They also showed that if triples contain at most one blank node the bound remains the same.

The common ground of these approaches is that they stick with the normative specification, that is, intensional RDFS. Other approaches such as RDF-F-Logic [13] depart from the normative specification. Finally, yet other approaches focus on the interplay between RDFS and other ontology languages such as OWL (e.g., RDFS(DL) [3]) and the family of description logics DL-Lite [2,1]. In contrast to the above approaches, our goal in this paper is to provide a bridge between the normative (intensional) and non normative (extensional) parts of the RDFS specification, and study systematically the latter.

6 Conclusions

In this paper we investigated the extensional semantics for RDFS. Based on the non-normative specification given in the standard W3C RDF semantics specification [7], we develop proof systems that show that one can get a practical, efficient and simple system for the extensional version of RDFS.

We answered an open problem since the publishing of the W3C RDF Semantics [7], which asked for the existence of a simple and efficient system of rules to codify extensional RDFS entailment. The results presented in the paper showed that providing a set of sound and complete inference rules for extensional RDFS is possible, and the complexity of computing the completion of an RDFS graph remains the same as in the normative case.

Our results will impact on current reasoning libraries (e.g., Jena) for RDFS that now can obtain more inferences at no significantly additional cost, as emphasized by our evaluation. Last, but not least, this extensional version aligns the semantics of RDFS and OWL, which previously were inconsistent due to the different meanings given by each of them to set-based notions such as subclass and subproperty.

Acknowledgments. We thank the anonymous referees that provided helpful suggestions. C. Gutierrez thanks the EMCL program for his stay at FUB. Franconi, Pirrò, Mosca and Gutierrez were supported by Marie Curie action IRSES - Net2 (Grant No. 24761). Gutierrez was supported by FONDECYT (Grant No. 1110287). R. Rosati was partially supported by the EU by FP7 project Optique – Scalable End-user Access to Big Data (Grant No. FP7-318338).

References

1. Artale, A., Calvanese, D., Kontchakov, R., Zakharyaschev, M.: The DL-Lite family and relations. J. Artif. Intell. Res. (JAIR) 36, 1–69 (2009)
2. Calvanese, D., Giacomo, G.D., Lembo, D., Lenzerini, M., Rosati, R.: Tractable reasoning and efficient query answering in description logics: The *DL-Lite* family. Journal of Automated Reasoning 39(3), 385–429 (2007)
3. Cuenca Grau, B.: A possible simplification of the semantic web architecture. In: WWW, pp. 704–713. ACM (2004)
4. De Bruijn, J., Franconi, E., Tessaris, S.: Logical reconstruction of normative RDF. In: OWL: Experiences and Directions Workshop (OWLED 2005), Galway, Ireland (2005)
5. Franconi, E., Gutierrez, C., Mosca, A., Pirrò, G., Rosati, R.: A Refutation System for Extensional RDFS. Technical report, KRDB, Free University of Bozen-Bolzano (2013), `http://www.inf.unibz.it/krdb/pub/tech-rep.php`
6. Gutierrez, C., Hurtado, C.A., Mendelzon, A.O., Pérez, J.: Foundations of semantic web databases. Journal of Computer and System Sciences 77(3), 520–541 (2011)

7. Hayes, P., McBride, B.: RDF semantics. W3C Reccomandation (2004), http://www.w3.org/tr/rdf-mt
8. Heath, T., Bizer, C.: Linked data: Evolving the web into a global data space. Synthesis Lectures on the Semantic Web: Theory and Technology 1(1), 1–136 (2011)
9. Klyne, G., Carroll, J.J., McBride, B.: Resource description framework (RDF): Concepts and abstract syntax. W3C Recommendation 10 (2004)
10. Marin, D.: A formalization of rdf. Technical report, Technical Report TR/DCC-2006-8, TR Dept. Computer Science, Universidad de Chile (2006)
11. Muñoz, S., Pérez, J., Gutierrez, C.: Simple and efficient minimal RDFS. Journal of Web Semantics 7(3), 220–234 (2009)
12. ter Horst, H.J.: Completeness, decidability and complexity of entailment for RDF schema and a semantic extension involving the OWL vocabulary. Web Semantics: Science, Services and Agents on the World Wide Web 3(2), 79–115 (2005)
13. Yang, G., Kifer, M.: Reasoning about anonymous resources and meta statements on the semantic web. In: Spaccapietra, S., March, S., Aberer, K. (eds.) Journal on Data Semantics I. LNCS, vol. 2800, pp. 69–97. Springer, Heidelberg (2003)

Appendix: Proof of Theorem 1

The following provides a sketch of the argument that proves the completeness of the $\vdash_{\rho\text{df}+}$ rule system: For graphs $\mathcal{G}$ and $\mathcal{H}$ in the ρdf+ vocabulary:

$$\mathcal{G} \vdash_{\rho\text{df}+} \mathcal{H} \text{ iff } \mathcal{G} \models_{\rho\text{df}+} \mathcal{H}.$$

While the soundness theorem (from left to right) follows straightforwardly from the observation that each rule in $\vdash_{\rho\text{df}+}$ preserves validity, the completeness theorem (from right to left) requires more effort to be proved. The proof is heavily based in the completeness theorem for the similar (intensional) $\vdash_{\rho\text{df}}$ system given in [11]. The notions of $\models_{\rho\text{df}}$ and $\vdash_{\rho\text{df}}$ can be found in that paper. First, we need some auxiliary notion of extended closure.

Definition 4. *The extended closure of a graph $\mathcal{G}$, denoted $\widehat{cl}(\mathcal{G})$, is the set of triples entailed from $\mathcal{G}$ under ρdf entailment ($\models_{\rho df}$) plus the axioms (14) - (17).*

We now rephrase $\widehat{cl}(\mathcal{G})$ using the $\vdash_{\rho\text{df}}$ rule system instead of $\models_{\rho\text{df}}$ entailment.

Lemma 1. *The* extended closure *of a graph $\mathcal{G}$ is the set of triples derived from $\mathcal{G}$ using $\vdash_{\rho df}$ plus the axioms (14) - (17).*

Proof. Use the known fact (Theorem 8 from [11]) that, if graphs $\mathcal{G}$ and $\mathcal{H}$ are in the ρdf vocabulary, $\mathcal{G} \vdash_{\rho\text{df}} \mathcal{H}$ iff $\mathcal{G} \models_{\rho\text{df}} \mathcal{H}$. □

The next lemma is at the key to the proof of the theorem:

Lemma 2 (Main). *If graphs $\mathcal{G}$ and $\mathcal{H}$ are in the ρdf vocabulary, then*

$$\widehat{cl}(\mathcal{G}) \vdash_{\rho df} \mathcal{H} \text{ iff } \mathcal{G} \vdash_{\rho df+} \mathcal{H}.$$

From Lemma 1 above it follows that we only have to show how each triple derived with the axioms (14) - (17) can be also derived with $\vdash_{\rho\text{df}+}$ and vice-versa.

The strategy aims at showing, through an *exhaustive combinatoric analysis*, that whatever can be derived by the axioms (14) to (17) can be derived with the $\vdash_{\rho\text{df}+}$ rule system as well. There are two operations working at the syntactic level: *axiom instantiation* and *pattern matching*. By means of these operations one can start combining together the axioms, until no more new syntactically well formed sentences are derivable. The proof strategy then is grounded on the fact that the only significant ways the axioms can be combined together give rise to nothing but the atoms that are present in the $\vdash_{\rho\text{df}+}$ system. Note that we can restrict to the case when $\mathcal{H}$ is one atom, because for ground atoms p, q it holds $\Sigma \models p \wedge q$ iff $\Sigma \models p$ and $\Sigma \models q$.

Proof. We will introduce for convenience auxiliary extended deductive rules allowing "implications" in the antecedent or in the consequent. This allows to codify formulas (14)-(17) as follows:

$$\text{14a}\ \frac{(A,\texttt{sc},B)}{(x,\texttt{type},A)\xrightarrow{\forall x}(x,\texttt{type},B)} \qquad \text{14b}\ \frac{(A,\texttt{sc},A)\wedge(B,\texttt{sc},B)\wedge(x,\texttt{type},A)\xrightarrow{\forall x}(x,\texttt{type},B)}{(A,\texttt{sc},B)} \qquad (\texttt{sc})$$

$$\text{15a}\ \frac{(P,\texttt{sp},Q)}{(x,P,y)\xrightarrow{\forall xy}(x,Q,y)} \qquad \text{15b}\ \frac{(A,\texttt{sc},A)\wedge(B,\texttt{sc},B)\wedge(x,P,y)\xrightarrow{\forall xy}(x,Q,y)}{(P,\texttt{sp},Q)} \qquad (\texttt{sp})$$

$$\text{16a}\ \frac{(P,\texttt{dom},A)}{(x,P,y)\xrightarrow{\forall xy}(x,\texttt{type},A)} \qquad \text{16b}\ \frac{(x,P,y)\xrightarrow{\forall xy}(x,\texttt{type},A)}{(P,\texttt{dom},A)} \qquad (\texttt{domain})$$

$$\text{17a}\ \frac{(P,\texttt{range},A)}{(x,P,y)\xrightarrow{\forall xy}(y,\texttt{type},A)} \qquad \text{17b}\ \frac{(x,P,y)\xrightarrow{\forall xy}(y,\texttt{type},A)}{(P,\texttt{range},A)} \qquad (\texttt{range})$$

The following are a few remarks to be made on the usage of this new system:

1. Rules with an implication in the antecedent (being universally quantified) cannot be fired from the graph $\mathcal{G}$ because of the presence of the *open world assumption*, we cannot know from $\mathcal{G}$ if it is valid or not.
2. Two implications can be matched if the meaning of the formulas allow so. For example, $(x,\texttt{type},A) \xrightarrow{\forall x} (x,\texttt{type},B)$ and $(y,\texttt{type},B) \xrightarrow{\forall y} (y,\texttt{type},C)$ would produce another rule:

$$\frac{(x,\texttt{type},A) \xrightarrow{\forall x} (x,\texttt{type},B) \qquad (y,\texttt{type},B) \xrightarrow{\forall y} (y,\texttt{type},C)}{(z,\texttt{type},A) \xrightarrow{\forall z} (z,\texttt{type},C)} \qquad (18)$$

3. The only way to use an implication in a combination of rules is, either:
   (a) To combine it with another implication to derive a third implication (e.g., to form rules of the form (18)). Table 4 summarizes the only admissible results one can obtain out the combination operation (we use the notation $r_1 \curvearrowright r_2$ to indicate that rule r_1 is combined with rule r_2). Note that the only possible relevant formula one could get with this procedure is a formula of the type $\forall x(x,\texttt{type},A) \rightarrow (x,\texttt{type},B)$, thus, to deduce a triple of the form $(u,\texttt{sc},v)$ using rule (14b). Note also that one cannot use the rules (15b), (16b) or (17b), because they need both variables universally quantified.

Table 3. Inference rules obtained by instantiating and combining rules (14a)-(17a). Rule 7bis can be obtained in turn from 7 and 6c, thus does not appear in Table 1.

Instantiation/Combination	Rule obtained	Rule in ρdf+	Rule in RDFS
(15a-inst)$\curvearrowright$16a$\curvearrowright$14b	$\frac{(\texttt{type},\texttt{sp},A),(A,\texttt{dom},B),(X,\texttt{sc},X)}{(X,\texttt{sc},B)}$	7	not available
(16a-inst)$\curvearrowright$14b$\curvearrowright$14a	$\frac{(\texttt{type},\texttt{dom},A),(X,\texttt{sc},X)}{(X,\texttt{sc},A)}$	7 bis	not available

(b) To instantiate the implication in the consequent, and using the Deduction Theorem ($p \vdash q \rightarrow r$ iff $p, q \vdash r$). Consider for instance rule (14a); we have: $(A, \texttt{sc}, B) \vdash (x, \texttt{type}, A) \xrightarrow{\forall x} (x, \texttt{type}, B)$. By using the deduction theorem, we obtain: $(A, \texttt{sc}, B)\ (x, \texttt{type}, A) \vdash (x, \texttt{type}, B)$. By systematically applying this process to rules (14a)-(17a), we obtain the rules in Table 5.

(c) To use instantiation that make it possible to combine rules. For example the new rule 7 Extensional follows from rule (15a) instantiated with $P = \texttt{type}$, which combined with the rule for domain (16a), gives the implication $\forall x(x, \texttt{type}, y) \rightarrow (x, \texttt{type}, B)$, which using rule (14b) gives $(y, \texttt{sc}, B)$ for y class. Table 3 shows the results of the application of the instantiation-plus-combination operation.

Table 4. Inference rules obtained by combining rules (14a)-(17a)

Combination	Rule obtained	Rule in $\vdash_{\rho\texttt{df+}}$	Rule in intensional RDFS
$14a\curvearrowright 14a$	$\frac{(A,\texttt{sc},B)\ \ (B,\texttt{sc},C)}{(A,\texttt{sc},C)}$	1b	rdfs 11
$15a\curvearrowright 15a$	$\frac{(P,\texttt{sp},Q)\ \ (Q,\texttt{sp},R)}{(P,\texttt{sp},R)}$	2b	rdfs 5
$15a\curvearrowright 16a$	$\frac{(P,\texttt{sp},Q)\ \ (Q,\texttt{dom},A)}{(P,\texttt{dom},A)}$	3b	not available
$15a\curvearrowright 17a$	$\frac{(P,\texttt{sp},Q)\ \ (Q,\texttt{range},A)}{(P,\texttt{range},A)}$	4b	not available
$16a\curvearrowright 14a$	$\frac{(P,\texttt{dom},A)\ \ (A,\texttt{sc},B)}{(P,\texttt{dom},B)}$	3c	not available
$17a\curvearrowright 14a$	$\frac{(P,\texttt{range},A)\ \ (A,\texttt{sc},B)}{(P,\texttt{range},B)}$	4c	not available

Table 5. Set of inference rules obtained by instantiating rules (14a)-(17a)

Rule Instantiated	Rule obtained	Rule in ρdf+	Rule in intensional RDFS
13a	$\frac{(A,\texttt{sc},B)\ \ (X,\texttt{type},A)}{(X,\texttt{type},B)}$	1a	rdfs 9
14a	$\frac{(P,\texttt{sp},Q)\ \ (X,P,Y)}{(X,Q,Y)}$	2a	rdfs 7
15a	$\frac{(P,\texttt{dom},A)\ \ (X,P,Y)}{(X,\texttt{type},A)}$	3a	rdfs 2
16a	$\frac{(A,\texttt{range},B)\ \ (X,A,Y)}{(Y,\texttt{type},B)}$	4a	rdfs 3

The presented proof system is the collection of all rules obtained. In particular, an exhaustive combinatorics indicates that *the only possible cases* are those considered in ρdf+. The idea is as follows:

1. Note that the only possible relevant formula one could get with the introduced procedure is a formula of the type $\forall x(x, \texttt{type}, A) \rightarrow (x, \texttt{type}, B)$, thus, to deduce a triple of the form $(u, \texttt{sc}, v)$ using rule (14b). Note that one cannot use the other rules (15b), (16b) or (17b), because they need both variables universally quantified.
2. With (1) in mind, one should start looking for the successful combinations.
   (a) Those that begin with $(x, \texttt{type}, y)$: could be rules (15a), (16a) or (17a) instantiated with $P = \texttt{type}$. As for Rule (15a), we should instantiate also $y = C$, but in this case the rule will give $\forall x(x, \texttt{type}, C) \rightarrow (x, Q, C)$, whose consequent cannot be further combined unless $Q = \texttt{type}$, which gives nothing. As for rule (16a), it gives our rule 7bis, while rule (17a) is useless for this argument (notice that in (17a) the y in the implication changes its position from third to first thus making impossible the combination with (14b)).
   (b) Those that end with $(x, \texttt{type}, y)$: here rule (16a) is relevant once y is instantiated to a constant; and rules (16a) and (17a) with the restriction $x = y$. It is not difficult to note that the first case is useful only for the instantiation $P = \texttt{type}$. In the second case, the only productive combination is to combine it with rule (15a) weakened to $x = y$. □

Now are read to prove the statement of Theorem 1:

Proof. $\mathcal{G} \models_{\rho\text{df}+} \mathcal{H}$
iff $\mathcal{G} \models_{\text{RDFS}+} \mathcal{H}$ (by definition of $\models_{\rho\text{df}+}$)
iff $\mathcal{G} \cup \{\text{axioms } 14 - 17\} \models_{\text{RDFS}} \mathcal{H}$ (by definition of RDFS+)
iff $\widehat{cl}(\mathcal{G}) \models_{\text{RDFS}} \mathcal{H}$ (by Definition 4)
iff $\widehat{cl}(\mathcal{G}) \models_{\rho\text{df}} \mathcal{H}$ (Theorem 5 from [11]) because left and right hand sides have only ρdf vocabulary)
iff $\widehat{cl}(\mathcal{G}) \vdash_{\rho\text{df}} \mathcal{H}$ (Soundness and completeness of ρdf –Theorem 8 from [11]– because there is only ρdf vocabulary)
iff $\mathcal{G} \vdash_{\rho\text{df}+} \mathcal{H}$ (by Lemma 2).

□

Indented Tree or Graph? A Usability Study of Ontology Visualization Techniques in the Context of Class Mapping Evaluation

Bo Fu[1], Natalya F. Noy[2], and Margaret-Anne Storey[1]

[1] Department of Computer Science, University of Victoria, BC, Canada
[2] Stanford Center for Biomedical Informatics Research, Stanford University, CA, US
{bofu,mstorey}@uvic.ca, noy@stanford.edu

Abstract. Research effort in ontology visualization has largely focused on developing new visualization techniques. At the same time, researchers have paid less attention to investigating the usability of common visualization techniques that many practitioners regularly use to visualize ontological data. In this paper, we focus on two popular ontology visualization techniques: indented tree and graph. We conduct a controlled usability study with an emphasis on the effectiveness, efficiency, workload and satisfaction of these visualization techniques in the context of assisting users during evaluation of ontology mappings. Findings from this study have revealed both strengths and weaknesses of each visualization technique. In particular, while the indented tree visualization is more organized and familiar to novice users, subjects found the graph visualization to be more controllable and intuitive without visual redundancy, particularly for ontologies with multiple inheritance.

Keywords: Ontology visualization, indented tree, graph, usability study.

1 Introduction

Information visualization (InfoVis) is a well-established research field. The goal of InfoVis is to transform information into visual representations that enable viewers to offload cognition to their perceptual systems in the process of better observing and understanding the information at hand. On the semantic web, researchers have applied visualization techniques to a range of topics such as semantic search [1], linked open data [2] and most notably, ontology design and management [3]. In recent years, ontology visualization has attracted much attention from the research community with a focus on providing the necessary support to enable users to create new and browse existing ontological resources. This research trend is reflected in the various visualization plugins developed for the Protégé[1] ontology editor [4], and visual support designed for querying and browsing ontology libraries [5, 6].

A commonly used technique in ontology visualization is indented tree where indentation is used to illustrate super/sub-class relationships and there is one path and

[1] http://protege.stanford.edu

H. Alani et al. (Eds.): ISWC 2013, Part I, LNCS 8218, pp. 117–134, 2013.

one path only between any pair of nodes. Another observation from the literature is that several ontology visualization tools have built upon node–link diagrams (i.e., graphs), which are essentially nodes with connecting edges that illustrate ontological entities and the relationships that exist among them. While researchers have devoted significant effort to develop new tools and techniques, they have paid less attention to investigating the usability of existing ontology visualization techniques that many practitioners already use on a regular basis.

Motivated by this research opportunity, we evaluated two frequently used ontology visualization techniques in the state of the art: indented tree and graph visualization. The goal of our study is to investigate the effectiveness and efficiency of the support that these visualization techniques provide. Specifically, we are interested in comparing their support to users during manual mapping evaluation tasks. We used a controlled experimental approach and present quantitative and qualitative analysis of the usability issues associated with these visualization techniques. The results from this research have uncovered useful information on the suitability of these visualization techniques in knowledge representation and mapping evaluation. In particular, we identified and highlighted perceived benefits and drawbacks of these techniques.

2 Related Work

In recent years, researchers have developed a variety of techniques to visualize ontologies. In this section, we present a brief overview on notable advances in this area. For extended discussions and classifications, see [3, 7].

Ontology development is one key activity that routinely relies on visualization. Visualizations assist users monitoring changes during ontology evolution [8], provide alternative development platforms by enabling UML-based editing [9] and rule-based authoring[2] of ontologies. Ontology editors such as Protégé, WebProtégé [10], OBO-Edit[3] and structOntology[4]; ontology browsers such as VectorBase[5]; ontology libraries such as BioPortal[6]; as well as ontology mapping tools such as OntoLink[7] all use indented tree visualization to present hierarchical structures that are typically associated with ontological entities. Others have applied treemaps [11] in ontology visualization to make use of all available screen space and to maximize the information displayed. Plaisant and colleagues [12] explored SpaceTree for ontology visualization, which extends the conventional node–link diagrams with dynamic rescaling to utilize screen space. Parsia and colleagues [13] proposed CropCircle, which illustrates parent–child and sibling relationships simultaneously. Protégé visualization plugins such as

[2] http://oogis.ru/component/option,com_remository/Itemid,34/func,fileinfo/id,15/lang,en see DroolsTab
[3] http://oboedit.org
[4] http://openstructs.org/structontology
[5] https://www.vectorbase.org/content/ontology-browser
[6] http://bioportal.bioontology.org
[7] http://www.mindswap.org/2004/OntoLink

OwlViz[8], NavigOWL[9], TGVizTab [19] and OWLPropViz[10] use graphs to illustrate classes and relationships in ontologies. Other web-based tools using similar node–link diagrams to visualize ontologies include FlexViz [20], BioMixer [21] and OLSVis [6]. In addition, 3D techniques have been applied to add more space on the screen by introducing a third dimension to node–link diagrams, such as OntoSphere [14]. Other research has focused on reducing information overload in node–link visualizations by presenting only classes above a calculated importance score [15], while several authors [16, 17, 18] have argued for the benefits of multiple visualizations with the goal of adapting to user preference and style.

The vast majority of these tools and techniques use indented tree or graph visualizations. Researchers in the field of InfoVis have extensively studied both techniques [22, 23] and proposed a range of evaluation approaches depending on the stage of the visualization software [24], including empirical studies [25] and insight-based methodologies [26]. In contrast, evaluation of visualization techniques in the context of ontology-focused tasks has been limited. Existing studies have compared Protégé plugins [27] and visualizations with built-in query support [28], focusing on evaluating their ability to support users seeking specific ontological information through controlled experiments. This paper aims to fill this research gap by presenting a comparative usability study of the commonly used indented tree and graph visualizations focusing on how well they illustrate ontological semantics.

3 Usability Study Overview

The goal of our usability study is to investigate the extent to which indented trees and graphs can support users in the process of understanding the semantics in ontologies.

Specifically, we asked the study participants to evaluate a given set of mappings between pairs of ontologies by interacting with the visualizations of these ontologies. To evaluate a mapping successfully, a participant must understand the semantics of the mapped entities in their respective ontologies and must use this knowledge to determine whether a mapping relation exists. Hence this task setup ensures the study focuses on examining the interactions between the participants and visualizations. Note that we did not explicitly specify that the participant must generate an overview of each ontology as we believe exploratory activities are inevitable in the given tasks. To generate a mapping correctly or to identify an incorrect one, the user typically must understand the semantics of the entities in their respective ontologies. This understanding is often a result of exploring the semantics and gaining an overview of the structures.

[8] http://www.co-ode.org/downloads/owlviz
[9] http://klatif.seecs.nust.edu.pk/navigowl
[10] http://www.wachsmann.tk/owlpropviz

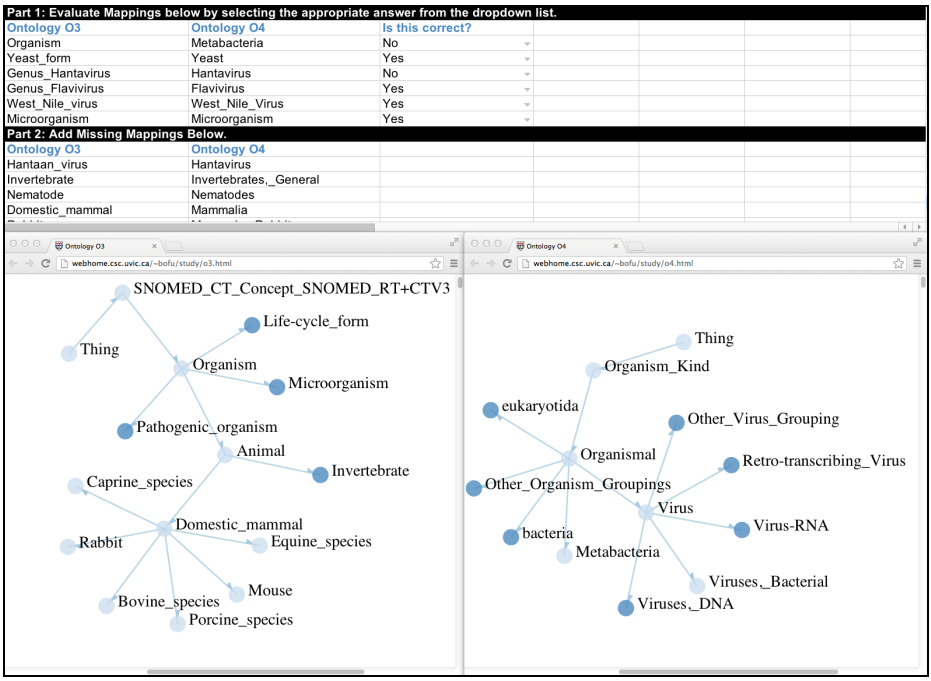


Fig. 1. Sample Task Screen. In this example, graphs are used to visualize two biomedical ontologies. Mappings to be evaluated are presented in a spreadsheet. Interacting with the visualizations, participants must use drop-down lists containing either "yes" or "no" responses to evaluate the correctness of existing mappings (in part 1) and add missing mappings by typing class names (into part 2 of the spreadsheet).

3.1 Tasks

We presented the participants with a set of mappings and asked them to identify correct and incorrect mappings as well as add missing mappings. The participants were assisted by visualizations of the ontology pair. Participants essentially engaged in two types of activities: identification activities and creation activities. The former involves the identification of correct and incorrect results among an existing set of mappings (i.e., determining the correctness of the given mappings). The latter involves the creation of new mappings that are absent from the existing set (i.e., determining the completeness of the given mappings). Figure 1 shows an example of what a participant saw on her (or his) screen.

3.2 Datasets

We used two pairs of ontologies, each accompanied by a set of mapping standards, taken from the Ontology Alignment Evaluation Initiative (OAEI) 2012 conference[11]

[11] http://oaei.ontologymatching.org/2012/conference/index.html

and the BioMed[12] tracks. Table 1 presents an overview of the ontologies used in this study.[13]

The conference ontologies describe the organization of conferences with a total of 74 and 100 classes respectively, at most 3 or 6 classes on the longest path to root, at most 8 or 9 subclasses for a class, without any multiple inheritance. The conference task represents a less difficult scenario, where the ontologies involved have fewer classes, the number of subclasses per class is fewer and the paths to root are shorter.

The BioMed task involves ontologies describing concepts in the organism domain. We reduced the size of the original ontologies and gold standards. In our study, the BioMed ontologies have a total of 89 and 181 entities respectively, at most 11 or 12 classes on the longest path to root, at most 6 or 10 subclasses for a class, with at most 4 occurrences of multiple inheritance. The BioMed task illustrates a more difficult scenario as the ontologies contain more entities, the number of children per entity is increased, the paths to root are longer and multiple inheritance is present.

Based on the OAEI gold standards, for each ontology pair, we randomly removed correct mappings from its gold standard and added incorrect mappings in order to create two mapping sets to present to the participants. The conference task and the BioMed task both required the participants to identify 13 correct results, 3 incorrect results and add 7 missing mappings in each task scenario. This setup thus ensures that the study outcome (in particular, time on task) is not affected by the number of mappings to be evaluated, but rather a result of ontology and visualization complexity.

Table 1. Characteristics of the Ontologies Used in the Study

	Conference Ontologies		BioMed Ontologies	
	O_1	O_2	O_3	O_4
Class Count	38	77	89	181
Object Property Count	13	33	-	-
Data Type Property Count	23	-	-	-
Multiple Inheritance Occurrences	-	-	2	4

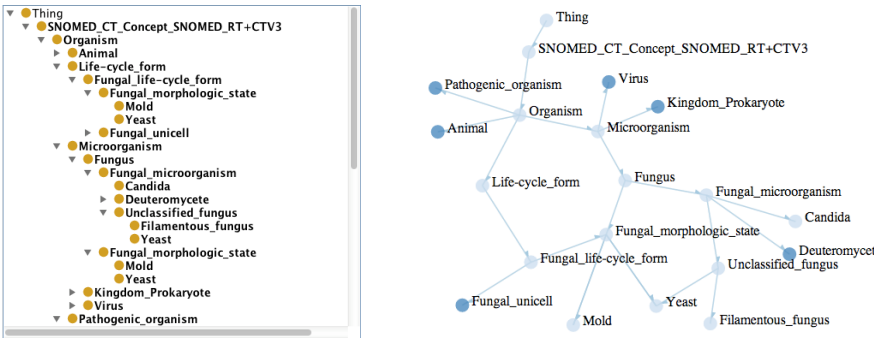
3.3 Visualization Support

We presented indented tree visualizations to the participants by loading ontologies into Protégé and asked participants to interact with the trees but not with any other features in Protégé. We implemented graph visualizations[14] in force directed layouts

[12] http://www.cs.ox.ac.uk/isg/projects/SEALS/oaei/2012/

[13] The ontologies and gold standard used in this study can be found at the prefix http://webhome.csc.uvic.ca/~bofu/study/datasets/ followed by file name: o1.owl, o2.owl, o3.owl, o4.owl, o1-o2%20gold%20standard.rdf or o3-o4%20gold%20standard.rdf

[14] The graph visualization of the ontologies used in the study can be found at the prefix http://webhome.csc.uvic.ca/~bofu/study/ followed by file name o1.html, o2.html, o3.html, or o4.html



(a) Indented Tree Visualization (b) Graph Visualization

Fig. 2. Visualization Techniques Investigated in the Study

using the D3 JavaScript library[15]. This implementation is representative of current graph techniques as it is composed of nodes and connecting edges, which are key characteristics of graphs as shown from the literature review.

Figure 2 presents visualization snippets of the SNOMED ontology using indented tree (Figure 2-a) and graph (Figure 2-b). In the indented tree visualization, *is-a* relationships are illustrated by indentation and the expanders allow users to toggle children of a node. Participants can use horizontal and vertical scroll bars to adjust the viewing area. In the graph visualization, classes are illustrated by vertices and *is-a* relationships are illustrated by directional edges with arrowheads pointing to the subclasses. The coloring of the vertices denotes whether a node is expandable (i.e., dark-colored vertices illustrate the existence of subclasses whereas light-colored vertices illustrate nonexpandable vertices). Clickable vertices allow users to toggle children of a particular node. In addition to using scroll bars to adjust the viewing area, the graph visualization is also editable: users can customize and manipulate the visualization by dragging and dropping nodes to any location on the screen. In both visualization techniques, we presented only the ontology root initially and participants must expand the root to view other classes.

3.4 Participants

We recruited volunteers via engineering departmental mailing lists at the University of Victoria. Each participant who successfully completed a study session received a $20 gift certificate. A total of 36 participants took part in our study. The participants were undergraduate and graduate students enrolled in disciplines including computer science, biomedical, biochemistry, and mechanical, electrical, software engineering. All participants were novice users of semantic technologies and they were new to ontologies and ontology mapping. As users of ontologies and visualizations

[15] http://d3js.org

increasingly include people with little knowledge of semantic technologies (e.g., BioPortal users are clinical and biomedical researchers who are new to ontologies and mappings), we were interested in studying the visualization support for novices. This group of users is of particular interest to us since a true expert should be able to successfully and accurately evaluate mappings regardless of the tool support. We do, however, recognize the opportunity to include expert users in future studies (discussed in section 6).

3.5 Protocol

We carried out one-on-one sessions with participants, where a session lasted approximately two hours. In each study session, we asked the participant to first complete an online tutorial on ontologies and ontology mapping.[16] The participant was then given instructions on the goal of her (or his) tasks: evaluate a set of exact mappings between a pair of ontologies.[17] We asked each participant to complete two tasks. Each task involved one ontology pair and one type of visualization. Each participant was asked to complete a video tutorial on how to interact with a given visualization before they began a task. We varied the ordering of the ontologies and visualization support between participants. For example, in one session, we asked Alice to complete the conference task using the indented tree, and then we asked her to complete the BioMed task using the graph. In another session, we asked Bob to complete the BioMed task using the indented tree, and then we asked him to complete the conference task using the graph. We randomly assigned tasks, ensuring equal distribution of tasks in the population as well as counterbalancing the order of tasks overall. This protocol ensured that a participant did not become overly familiar with a particular visualization, nor did the participant learn about the domain of interest over time, thus minimizing the impact of task order on the study outcome. Our protocol ensures that the only independent variable in the experiment is the visualization type, since we are interested in how two visualizations differ in their support to the same user group conducting the same set of tasks. However, we recognize a potential research opportunity to compare behaviors of different user groups in the future (discussed in section 6).

3.6 Metrics

To investigate the extent to which the indented tree and graph visualization can assist novice users in evaluating mappings *effectively* and *efficiently*, we measured task success and time on task as follows.

We calculated success scores for a participant to reflect *identification* success (i.e., the activity focusing on evaluating the correctness), *creation* success (i.e., the activity focusing on evaluating the completeness) and *overall* success (i.e., combing both type

[16] Materials used in this study can be found at `https://sites.google.com/site/uvicstudy`

[17] Since the evaluation process does not differ regardless of the type of entity or mapping relation, it is thus sufficient to use exact mappings as examples for the purpose of this study.

of activities). For example, suppose a task presents a set of existing mappings between ontology O and O', among which are n_1 number of correct mappings, n_2 number of incorrect mappings and n_3 number of missing mappings. If a participant successfully identifies x number of correct mappings and y number of incorrect mappings, then her (or his) identification success = $(x+y)/(n_1+n_2)$. If a participant correctly creates z number of new mappings, then her (or his) creation success = z/n_3. Her (or his) overall success = $(x+y+z)/(n_1+n_2+n_3)$. Her (or his) error rate is recorded as the number of incorrect answers divided by her (or his) total number of answers. Success scores range between 0 and 1; the higher the score the more successful the participant was at the task. Error rates also range between 0 and 1; the lower it is, the fewer mistakes the participant made in the task.

We asked participants to raise any questions before they began a task as we did not allow any interactions during the task. This restriction ensured a clear end state in the tasks, whereby time on task is the length of time it took a participant to complete the spreadsheet (which included both identification and creation activities).

3.7 Participant Feedback

After each task, we collected user feedback through computerized surveys based on the NASA-task load index (NASA-TLX) [29], the System Usability Scale (SUS) [30], the Usefulness, Satisfaction and Ease of Use (USE) questionnaire [31], and reaction cards [32]. We used 7-point Likert scales for all questionnaires.

Workload is "the cost of accomplishing mission requirements for the human operator" [33]. The NASA-TLX is specifically designed to measure workload through six dimensions, namely mental demand, physical demand, temporal demand, effort, performance and frustration level. Each dimension is measured through a question that asks the participant to rate the demand level on scales with endpoints being low-high and poor-good. In this study, we used raw NASA-TLX [34], which eliminates weightings between paired dimensions. Raw NASA-TLX is shown to be of no particular accuracy loss compared to the original, weighting NASA-TLX [33]. For each participant, we calculated a single workload score by averaging normalized scores of the six dimensions. The workload rating for a dimension ranges between 0 and n-1 given an n-point Likert scale. The workload score for a dimension is calculated as (n-1)×100/6. The overall workload is the mean of six ratings. The workload rating ranges between 0 (low workload) and 100 (high workload).

The SUS is a questionnaire that contains 10 statements collecting feedback on agreement scales. Five statements are positively worded and the other five are negatively worded. Example statements include "I thought the visualization was easy to use" and "I found the visualization unnecessarily complex". Using an n-point Likert scale, the score contribution for a statement ranges between 0 and n-1. For a positively worded statement, the score contribution is the scale position minus 1. For a negatively worded statement, the score contribution is n minus the scale position. Multiply the sum of ten score contributions by 10/(n-1) to obtain the overall usability score. An aggregated usability score can be calculated for a visualization, which ranges between 0 and 100. The higher it is, the more usable the visualization.

In addition to SUS, we used the USE questionnaire, which expresses usability in four dimensions: usefulness, ease of use, ease of learning and satisfaction. We collected levels of user agreement to 30 statements. Our goals was to gain a further understanding in the variations between the two visualization techniques by breaking usability down to four dimensions. We calculated a mean rating that ranges between 0 and 6 to indicate the average rating for each of the four usability dimensions.

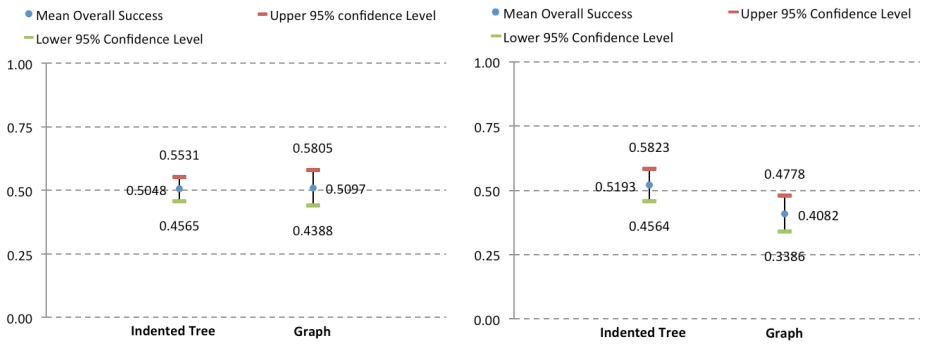
Finally, we presented 118 reaction cards containing adjectives (e.g., "engaging", "powerful", "rigid", "dated", etc.) to participants after the completion of each task. We asked the participants to pick out top five cards that best described the specific visualization in the given task and explain their choices. This technique aimed to elicit commentary and collect qualitative feedback.

4 Findings

We present the results of the measures discussed in section 3.6 and 3.7 below.

4.1 Effectiveness

Figure 3 presents mean overall success. Table 2 presents further details on the various success scores. In the conference task, the user group that was assisted by graphs yielded a slightly higher mean overall success score. Both visualization techniques generated the same median overall success score. We carried out independent sample t tests (with an alpha level equal to 0.05) with the null hypothesis being there is no difference between the two user groups. P-values from these independent t tests indicate that there is no significant difference between the user groups.



(a) Conference Task (b) BioMed Task

Fig. 3. Visualization Effectiveness. The vertical axis illustrates mean overall success and the horizontal axis represents the user groups using different visualizations. Error bars show 95% confidence intervals, i.e., how far from the reported value the true (error free) value might be.

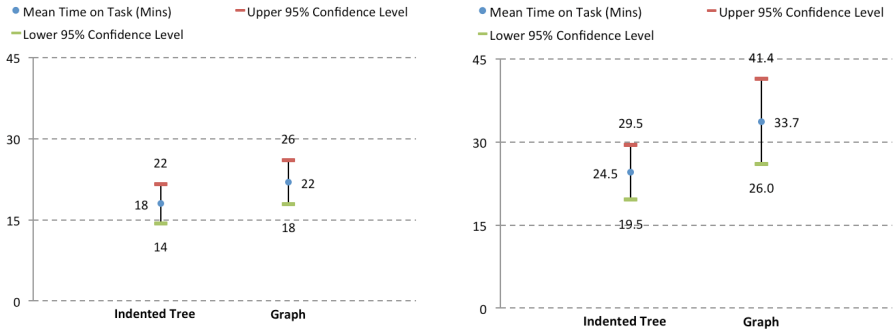
In the BioMed task, the user group who used indented trees generated higher mean, median and lower standard deviation in identification and overall success scores. They also made fewer mistakes, as suggested by lower and less dispersed error rates. With the exception of creation success scores, p-values generated from all other scores are equal to or less than the alpha level. This finding suggests that there are significant differences between the identification score, overall success and error rates between the two user groups, indicating that indented trees were more effective.

Table 2. Task Success. Statistically significant results are bolded.

Visualization	Task Success	Conference Task			BioMed Task		
		Mean	Median	StDev	Mean	Median	StDev
Indented Tree	Identification	0.6458	0.6250	0.1134	**0.6944**	**0.6875**	**0.1514**
	Creation	0.1825	0.1429	0.2068	0.1190	0	0.1715
	Overall	0.5048	0.5000	0.1045	**0.5193**	**0.5000**	**0.1362**
	Error	0.3951	0.4045	0.1322	**0.3668**	**0.3640**	**0.1433**
Graph	Identification	0.6563	0.6563	0.1458	**0.5382**	**0.5625**	**0.2013**
	Creation	0.1746	0.0714	0.2525	0.1111	0.0714	0.1433
	Overall	0.5097	0.5000	0.1534	**0.4082**	**0.4130**	**0.1507**
	Error	0.3794	0.3787	0.1339	**0.4747**	**0.5147**	**0.1747**

4.2 Efficiency

Figure 4 presents an overview of the average time spent completing each task using different visualizations. Further details are shown in Table 3. It is consistently shown in both tasks that user groups assisted by the indented tree visualization were faster at completing their tasks than those who used graphs. However, p-values do not provide sufficient evidence to indicate a statistically significant difference between the user groups, suggesting comparable completion time regardless of the visualization used.



(a) Conference Task (b) BioMed Task

Fig. 4. Visualization Efficiency. The vertical axis represents mean time-on-task, and the horizontal axis illustrates the user group. Error bars show 95% confidence intervals.

Table 3. Time on Task

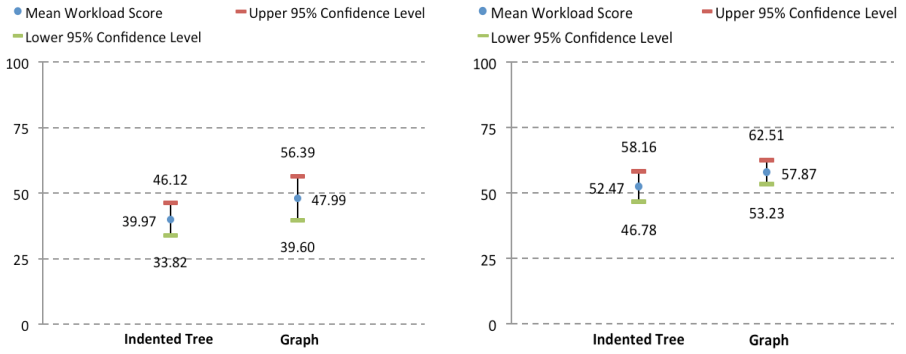
Visualization	Conference Task			BioMed Task		
	Mean	Median	StDev	Mean	Median	StDev
Indented Tree	17.9	15.5	7.8	24.5	25	10.7
Graph	21.9	21	8.9	33.7	30	16.6

4.3 Workload

Figure 5 presents an overview of the workload scores. Further details are presented in Table 4. Mean values indicate that the user group assisted by the graph visualization found the task more demanding that those who used the indented tree visualization in both tasks. In the conference task, the workload scores for graphs are particularly disperse, which consequently led to higher mean and median even though the most common rating is much lower (see mode) compared to the indented tree visualization. However, p-values in both tasks indicate that the differences between the two user groups are not statistically significant, i.e., there was no particular increase in workload regardless of the type of visualization used.

Table 4. Workload Scores

Visualization	Conference Task				BioMed Task			
	Mean	Median	Mode	StDev	Mean	Median	Mode	StDev
Indented Tree	39.97	41.67	50.00	13.31	52.47	52.78	52.78	12.31
Graph	47.99	45.83	30.56	18.17	57.87	56.97	61.11	10.05

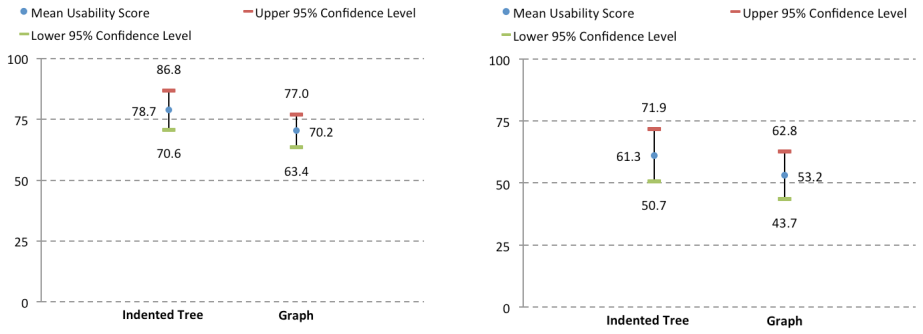


(a) Conference Task (b) BioMed Task

Fig. 5. Task Workload. The vertical axis represents mean workload scores, and the horizontal axis illustrates the user group. Error bars show 95% confidence intervals.

4.4 Usability and Qualitative Feedback

Figure 6 presents an overview of the SUS scores. Further details are shown in Table 5. In both tasks, the usability scores indicate that participants found the indented tree more usable than graph as the average, mid-point and most commonly occurred values are always higher. However, this difference is not statistically significant.



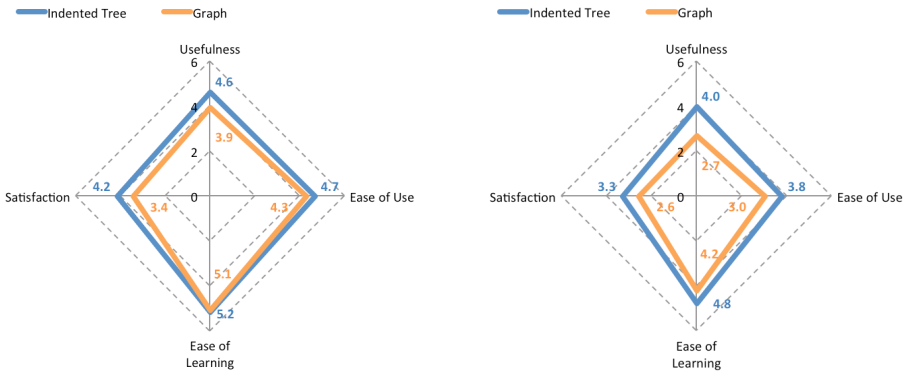
(a) Conference Task (b) BioMed Task

Fig. 6. Visualization Usability. The vertical axis represents mean usability score, and the horizontal axis illustrates the user group. Error bars show 95% confidence intervals.

Figure 7 shows USE ratings generated for the visualizations. In the conference task, similar ratings are generated for both visualization techniques with the indented tree having slightly higher mean and median ratings in all four dimensions. However, p-values suggest that the differences shown in this task are not statistically significant. In the BioMed task, there is a decrease in all ratings for both visualization techniques, although higher mean and median values are found in indented tree. P-values indicate a statistically significant difference between the two visualization techniques in terms of usefulness (note that statistical significance was not found in ease of use, ease of learning and satisfaction ratings). The results suggest that as the evaluation task becomes more difficult, visualization support appears to be less helpful regardless of the specific technique. Overall, the USE results indicate that all usability dimensions of the two visualization techniques are in fact very comparable.

Table 5. Usability Scores

Visualization	Conference Task				BioMed Task			
	Mean	Median	Mode	StDev	Mean	Median	Mode	StDev
Indented Tree	78.70	80.00	71.67	17.54	61.30	60.00	70.00	22.96
Graph	70.19	73.33	50.00	14.72	53.24	50.83	50.00	20.67



(a) Conference Task (b) BioMed Task

Fig. 7. Visualization Usability Breakdown. Each axis presents a usability dimension (usefulness, ease of use, ease of learning and satisfaction). The radar chart presents a mean rating for each dimension.

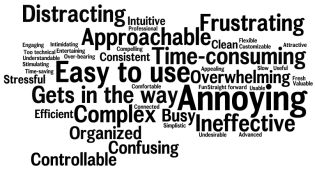
Visualization	Conference Task	BioMed Task
Indented Tree		
Graph		

Fig. 8. Reaction Card Responses. Font sizes illustrate the frequency of use for a particular card; the bigger the font, the more frequently the card was used to describe a visualization.

Figure 8 presents tag clouds of reaction card responses. In the conference task, participants found both visualization techniques easy to use. They found the indented tree *familiar* and the graph *intuitive*. In the case of the BioMed task, as the task becomes more difficult, more diverse reaction cards are used and an increased number of negative cards are present for both visualization techniques. For instance, the participants described both visualization techniques as *distracting*, *frustrating* and *confusing*. They characterized indented trees as *organized*, *straightforward* and *simplistic*, although *dull*, *boring* and *busy*. They found the graph visualization to be *approachable* and *controllable* in the conference task, however, it became *annoying* and *complex* in the BioMed task. Overall, participants consistently used *simplistic* to describe the

indented tree in both tasks. They also consistently used *easy to use* to describe the graph visualization in both tasks, although this phrase is used much less frequently to describe the indented tree in the BioMed task. Furthermore, several participants mentioned that they particularly liked how multiple inheritance is visualized in graphs.

5 Discussion

We present correlation results and key observations drawn from this study next.

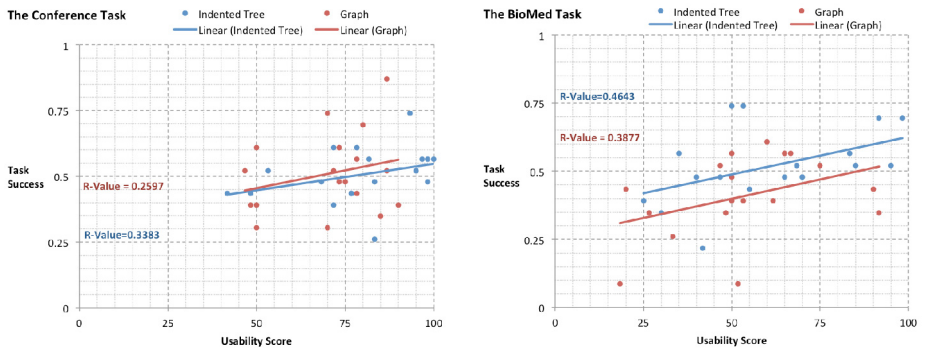
5.1 Correlation Tests

Given the range of variables (i.e., effectiveness, efficiency, workload, SUS and USE scores) associated with each visualization, we conducted correlation tests to determine whether dependable relationships exist. If a strong correlation coefficient exists between a variable pair, then knowing the value of one variable, we could predict the likely value of the other variable for a given visualization. The degree of correlation between two variables is represented by the R-value, which ranges between -1 and 1. The stronger the correlation, the closer the R-value is towards -1 (negative correlation) or 1 (positive correlation). Overall, results show that R-values indicate mostly weak or non-existent associations between variables. An example is presented in Figure 9. In Figure 9-a, task success is correlated with usability scores. R-values indicate that visualization usability did not impact task success. In Figure 9-b, error rates are correlated with task completion time. Notably in the BioMed task, we found a stronger R-value suggesting that if more time is spent to complete a task, users using graphs are likely to make fewer mistakes.

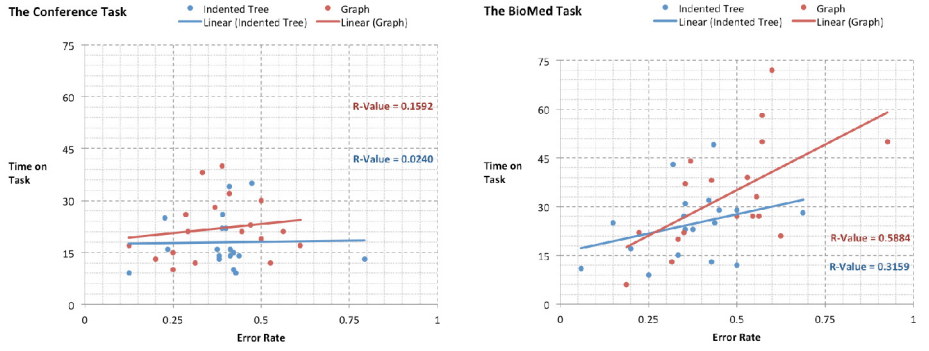
5.2 Summary of Findings

The effectiveness results suggest that when ontologies are smaller and have a simpler structure, users are likely to achieve the same level of success regardless of the specific visualization used, such as the case with the conference task. However, given more complex ontologies, the indented tree is more effective. More specifically, users are likely to be more successful at activities that concern the evaluation of existing mappings using indented trees, but more successful at activities that involve creating new mappings using the graphs. This finding suggests that the indented tree visualization is more suitable for list-checking activities, and the graph visualization is more suitable for overviews.

The efficiency results suggest that the task completion time is more likely to be a result of domain familiarity (the majority of participants being engineering students) rather than a direct cause of the specific visualization used, since both tasks had comparable completion times and the differences are not statistically significant. Similar findings are shown in the workload ratings, where participants did not feel a particular visualization is more demanding than the other.



(a) Correlating Task Success and Usability Score



(b) Correlating Error Rate and Time on Task

Fig. 9. Correlation Results. The axes represent the variables being tested for correlation. The scattered plots illustrate individual participant results. Trend lines indicate linear regressions between the variables.

Another notable finding is that although most participants were interacting with graphs for the first time, they did not feel that it was more difficult to learn as suggested by the ease of learning ratings. Some mentioned that graphs held their attention better. However, it is clear that graphs can become difficult to manage once they exceeded a certain threshold of nodes. This finding suggests that the graph size should not be overlooked when determining the suitability of its application.

Multiple inheritance is inevitable in certain domains. Visualization techniques that can seamlessly incorporate such conceptual models are essential to users, and as such, graphs are more suitable than indented trees in these scenarios as noted by several participants. Take the SNOMED visualization snippets shown in Figure 2 as an example. *Yeast* has two parents: *Fungal_morphologic_state* and *Unclassified_fungus*; *Fungal_morphologic_state* also has two parents: *Fungal_life-cycle_form* and *Fungus*. This semantic structure is illustrated with ease using directional edges in the graph visualization (see Figure 2-a). However, *Fungal_morphologic_state* is shown twice and *Yeast* appears three times (see Figure 1-b) in the indented tree. This visual duplication in indented trees requires users to make additional efforts when understanding the data at hand and can potentially add to confusion.

Another disadvantage of the indented tree is that given a fixed screen space, it is not always possible to view the entire tree structure. It is particularly challenging given ontologies with greater depth and a large number of descendants per node. The sheer amount of expanders can be overwhelming and this makes it difficult for the user to preserve a mental model of the ontological hierarchy. While the indented tree offers little adaptation to the user, the graph visualization is much more customizable and adaptive. For example, users can simply place previously explored nodes on the far side of the screen to make room for nodes that are of current interest. Users stated that the flexibility offered by graphs helped them to better hold their attention during the tasks. A disadvantage of the graph is that it can quickly get busy on a fixed screen size providing ineffective visualization given a large number of nodes. Overall, the advantage of the indented tree is that it is familiar and predictable, as most participants are already accustomed to this visualization technique given its similarity with computer file directories. However, we attempted to minimize this bias by presenting visualizations in Protégé (given it is representative of state-of-the-art indented tree techniques and none of the participants have encountered it before), as it is unlikely for one to find participants who have never seen a computer directory before participating in our experiment.

6 Conclusions, Limitations and Future Work

Given the different strengths and weaknesses associated with graphs and indented trees, their applications should thus be determined upon specific ontology characteristics, visualization needs and user goals. Tool designers should consider combining multiple ontology visualization techniques that can engage users from different viewpoints yet are complementary to one another. In addition, ontology visualization should aim to empower users by providing customizable visualizations that are not only in manageable segments but are also adaptive to diverse personal preferences and styles.

The results of our study are dependent upon the visualization implementation, datasets used and participants involved. Although the graph visualization is representative of current techniques, some behaviors are unique to this specific force directed implementation. For instance, class names can overlap in graphs, and although participants can easily drag and rearrange nodes for a better view of the text, this process can increase frustration for users. Nevertheless, we have uncovered some motivating results from this study.

Our study suggests several future research directions. First, it would be useful to conduct studies with larger participant groups, as increased sample sizes could potentially lead to more statistically significant findings. Feedback regarding the controllable nature of graphs is specific to the implementation used in this study. Future experiments could explore non-editable graph layouts as well as other visualization techniques, such as treemaps and SpaceTrees. In addition, although it is relevant to investigate usability issues that arise among novice users, it can be even more informative for the study to recruit true ontology and mapping experts. Secondly, the datasets used in this study involve a limited set of ontologies and mappings. Future studies including larger ontologies from other domains and an increased number of mappings may uncover additional scalability issues. However, it may be challenging to recruit volunteers given tasks that could take hours or days to complete. Moreover,

the ontologies used in this study mostly contain hierarchical relationships among classes. Other object properties (e.g., transitive relationships, inverse relationships, etc.) associated with ontological entities can be the focus of further studies. For instance, future experiments could investigate whether graphs are more suitable to visualize object properties. Lastly, it may be beneficial to apply other evaluation approaches discussed in section 2 such as identifying usability issues based on observations of users over a long period of time.

Acknowledgment. This research is supported by the National Center for Biomedical Ontology (NCBO) under grant U54 HG004028 from the National Institutes of Health.

References

[1] Stab, C., Nazemi, K., Breyer, M., Burkhardt, D., Kohlhammer, J.: Semantics Visualization for Fostering Search Result Comprehension. In: Simperl, E., Cimiano, P., Polleres, A., Corcho, O., Presutti, V. (eds.) ESWC 2012. LNCS, vol. 7295, pp. 633–646. Springer, Heidelberg (2012)

[2] Dadzie, A.S., Rowe, M.: Approaches to Visualising Linked Data: A Survey. Semantic Web 2, 89–124 (2011)

[3] Katifori, A., Halatsis, C., Lepouras, G., Vassilakis, C., Giannopoulou, E.: Ontology Visualization Methods - A Survey. ACM Computing Surveys 39(4), Article 10 (2007)

[4] Sivakumar, R., Arivoli, P.V.: Ontology Visualization Protégé Tools – A Review. International Journal of Advanced Information Technology 1(4) (2011)

[5] Noy, N.F., Shah, N.H., Whetzel, P.L., Dai, B., Dorf, M., Griffith, N., Jonquet, C., Rubin, D.L., Storey, M.-A., Chute, C.G., Musen, M.A.: BioPortal: Ontologies and Integrated Data Resources at the Click of A Mouse. Nucleic Acids Research 37(2), 170–173 (2009)

[6] Vercruysse, S., Venkatesan, A., Kuiper, M.: OLSVis: An Animated, Interactive Visual Browser for Bio-ontologies. BMC Bioinformatics 13(1), 116 (2012)

[7] Lanzenberger, M., Sampson, J., Rester, M.: Ontology Visualization: Tools and Techniques for Visual Representation of Semi-Structured Meta-Data. Journal of Universal Computer Science 16(7), 1036–1054 (2010)

[8] Khattak, A.M., Latif, K., Khan, S., Ahmed, N.: Ontology Recovery and Visualization. In: The 4th International Conference on Next Generation Web Services Practices, pp. 90–96 (2008)

[9] Brockmans, S., Volz, R., Eberhart, A., Löffler, P.: Visual Modeling of OWL DL Ontologies using UML. In: McIlraith, S.A., Plexousakis, D., van Harmelen, F. (eds.) ISWC 2004. LNCS, vol. 3298, pp. 198–213. Springer, Heidelberg (2004)

[10] Tudorache, T., Nyulas, C., Noy, N.F., Musen, M.A.: WebProtege: A Collaborative Ontology Editor and Knowledge Acquisition Tool for the Web. Semantic Web Journal 4(1), 89–99 (2013)

[11] Baehrecke, E.H., Dang, N., Babaria, K., Shneiderman, B.: Visualization and Analysis of Microarray and Gene Ontology Data with Treemaps. BMC Bioinformatics 5, 84 (2004)

[12] Plaisant, C., Grosjean, J., Bederson, B.B.: SpaceTree: Supporting Exploration in Large Node Link Tree, Design Evolution and Empirical Evaluation. In: The IEEE Symposium on Information Visualization, vol. 57 (2002)

[13] Parsia, B., Wang, T., Goldbeck, J.: Visualizing Web Ontologies with CropCircles. In: The 4th International Semantic Web Conference. LNCS, vol. 3729, pp. 6–10 (2005)

[14] Bosca, A., Bomino, D., Pellegrino, P.: OntoSphere: More than A 3D Ontology Visualization Tool. In: The 2nd Italian Semantic Web Workshop. CEUR-WS, vol. 166 (2005)

[15] Motta, E., Mulholland, P., Peroni, S., d'Aquin, M., Gomez-Perez, J.M., Mendez, V., Zablith, F.: A Novel Approach to Visualizing and Navigating Ontologies. In: Aroyo, L., Welty, C., Alani, H., Taylor, J., Bernstein, A., Kagal, L., Noy, N., Blomqvist, E. (eds.) ISWC 2011, Part I. LNCS, vol. 7031, pp. 470–486. Springer, Heidelberg (2011)
[16] Storey, M.-A., Musen, M., Silva, J., Best, C., Ernst, N., Fergerson, R., Noy, N.: Jambalaya: Interactive Visualization to enhance Ontology Authoring and Knowledge Acquisition in Protégé. In: Workshop on Interactive Tools for Knowledge Capture (2001)
[17] Petrelli, D., Mazumdar, S., Dadzie, A.S., Ciravegna, F.: Multi Visualization and Dynamic Query for Effective Exploration of Semantic Data. In: Bernstein, A., Karger, D.R., Heath, T., Feigenbaum, L., Maynard, D., Motta, E., Thirunarayan, K. (eds.) ISWC 2009. LNCS, vol. 5823, pp. 505–520. Springer, Heidelberg (2009)
[18] Kuhar, S., Podgorelec, V.: Ontology Visualization for Domain Experts: A New Solution. In: The 16th International Conference on Information Visualisation, pp. 363–369 (2012)
[19] Alani, H.: TGVizTab: An Ontology Visualisation Extension for Protégé. In: Workshop on Visualization Information in Knowledge Engineering (2003)
[20] Falconer, S.M., Callendar, C., Storey, M.-A.: A Visualization Service for the Semantic Web. In: Cimiano, P., Pinto, H.S. (eds.) EKAW 2010. LNCS, vol. 6317, pp. 554–564. Springer, Heidelberg (2010)
[21] Fu, B., Grammel, L., Storey, M.-A.: BioMixer: A Web-based Collaborative Ontology Visualization Tool. In: The 3rd International Conference on Biomedical Ontology. CEUR-WS, vol. 897 (2012) ISSN 1613-0073
[22] Herman, I., Melançon, G., Marshall, M.S.: Graph Visualization and Navigation in Information Visualization: A Survey. IEEE Transactions on Visualization and Computer Graphics 6(1), 24–43 (2000)
[23] Graham, M., Kennedy, J.: A Survey of Multiple Tree Visualisation. Information Visualization 9(4), 235–252 (2009)
[24] Lam, H., Bertini, E., Isenberg, P., Plaisant, C., Carpendale, S.: Seven Guiding Scenarios for Information Visualization Evaluation. Technical Report (2010)
[25] Chen, C., Czerwinski, M.: Empirical Evaluation of Information Visualizations: An Introduction. International Journal on Human-Computer Studies 53, 631–635 (2000)
[26] Saraiya, P., North, C., Duca, K.: An Insight-Based Methodology for Evaluating Bioinformatics Visualization. IEEE Trans. on Visualization and Computer Graphics 11(4) (2005)
[27] Akrivi, K., Elena, T., Constantin, H., Georgios, L., Costas, V.: A Comparative Study of Four Ontology Visualization Techniques in Protégé: Experiment Setup and Preliminary Results. In: The 10th International Conference on Information Visualization, pp. 417–423 (2006)
[28] Swaminathan, V., Sivakumar, R.: A Comparative Study of Recent Ontology Visualization Tools with a Case of Diabetes Data. International Journal of Research in Computer Science 2(3), 31–36 (2012)
[29] Hart, S.G., Staveland, L.E.: Development of NASA-TLX (Task Load Index): Results of Empirical and Theoretical Research. Human Mental Workload. North Holland Press, Amsterdam (1988)
[30] Brooke, J.: SUS: A Quick and Dirty Usability Scale. Usability Evaluation in Industry. Taylor & Francis, London (1996)
[31] Lund, A.: Measuring Usability with the USE Questionnaire. Usability and User Experience Newsletter of the STC Usability SIG (2001)
[32] Benedek, J., Miner, T.: Measuring Desirability: New Methods for Evaluating Desirability in a Usability Lab Setting. In: Usability Professionals Association Conference (2002)
[33] Hart, S.G.: NASA-Task Load Index (NASA-TLX); 20 Years Later. In: The Human Factors and Ergonomics Society 50th Annual Meeting, pp. 904–908 (2006)
[34] Likert, R.: A Technique for the Measurement of Attitudes. Archives of Psychology 140, 1–55 (1932)

Real-Time RDF Extraction from Unstructured Data Streams

Daniel Gerber, Sebastian Hellmann, Lorenz Bühmann, Tommaso Soru, Ricardo Usbeck, and Axel-Cyrille Ngonga Ngomo

Universität Leipzig, Institut für Informatik, AKSW,
Postfach 100920, D-04009 Leipzig, Germany
{dgerber,hellmann,buehmann,tsoru,usbeck,ngonga}@informatik.uni-leipzig.de
http://aksw.org

Abstract. The vision behind the Web of Data is to extend the current document-oriented Web with machine-readable facts and structured data, thus creating a representation of general knowledge. However, most of the Web of Data is limited to being a large compendium of encyclopedic knowledge describing entities. A huge challenge, the timely and massive extraction of RDF facts from unstructured data, has remained open so far. The availability of such knowledge on the Web of Data would provide significant benefits to manifold applications including news retrieval, sentiment analysis and business intelligence. In this paper, we address the problem of the actuality of the Web of Data by presenting an approach that allows extracting RDF triples from unstructured data streams. We employ statistical methods in combination with deduplication, disambiguation and unsupervised as well as supervised machine learning techniques to create a knowledge base that reflects the content of the input streams. We evaluate a sample of the RDF we generate against a large corpus of news streams and show that we achieve a precision of more than 85%.

1 Introduction

Implementing the original vision behind the Semantic Web requires the provision of a Web of Data which delivers timely data at all times. The foundational example presented in Berners-Lee et al's seminal paper on the Semantic Web [3] describes a software agent who is tasked to find medical doctors with a rating of excellent or very good within 20 miles of a given location at a given point in time. This requires having timely information on which doctors can be found within 20 miles of a particular location at a given time as well as having explicit data on the rating of said medical doctors. Even stronger timeliness requirements apply in decision support, where software agents help humans to decide on critical issues such as whether to buy stock or not or even how to plan their drive through urban centers. Furthermore, knowledge bases in the Linked Open Data (LOD) cloud would be unable to answer queries such as "Give me all news of the last week from the New York Times pertaining to the director of a company".

H. Alani et al. (Eds.): ISWC 2013, Part I, LNCS 8218, pp. 135–150, 2013.

Although the current LOD cloud has tremendously grown over the last years [1], it delivers mostly encyclopedic information (such as albums, places, kings, etc.) and fails to provide up-to-date information that would allow addressing the information needs described in the examples above.

The idea which underlies our work is thus to alleviate this current drawback of the Web of Data by developing an approach that allows extracting RDF from unstructured (i.e., textual) data streams in a fashion similar to the live versions of the DBpedia[1] and LinkedGeoData[2] datasets. The main difference is yet that instead of relying exclusively on structured data like LinkedGeoData or on semi-structured data like DBpedia, we rely mostly on unstructured, textual data to generate RDF. By these means, we are able to unlock some of the potential of the document Web, of which up to 85% is unstructured [8]. To achieve this goal, our approach, dubbed RdfLiveNews, assumes that it is given unstructured data streams as input. These are deduplicated and then used as basis to extract patterns for relations between known resources. The patterns are then clustered to labeled relations which are finally used as basis for generating RDF triples. We evaluate our approach against a sample of the RDF triples we extracted from RSS feeds and show that we achieve a very high precision.

The remainder of this work is structured as follows: We first give an overview of our approach and give detailed insights in the different steps from unstructured data streams to RDF. Then, we evaluate our approach in several settings. We then contrast our approach with the state of the art and finally conclude.

2 Overview

We implemented the general architecture of our approach dubbed RdfLiveNews according to the pipeline depicted in Figure 1. First, we gather textual data from data streams by using RSS feeds of news articles. Our approach can yet be employed on any unstructured data published by a stream. Since input streams from the Web can be highly redundant (i.e., convey the same information), we then deduplicate the set of streams gathered by our approach. Subsequently, we apply a pattern search to find lexical patterns for relations expressed in the text. After a refinement step with background knowledge, we finally cluster the extracted patterns according to their semantic similarity and transform this information into RDF.

2.1 Data Acquisition

Formally, our approach aims to process the output of unstructured data sources S^i by continuously gathering the data streams D^i that they generate. Each data stream consists of atomic elements d^i_j (in our case sentences). Let $D^i_{[t,t+d]}$ be the portion of D^i that was emitted by S^i between the times t and $t+d$. The data gathering begins by iteratively gathering the elements of the streams $D^i_{[t,t+d]}$. from all available sources S^i for a period of time d, which we call the *time*

[1] `http://live.dbpedia.org/sparql`
[2] `http://live.linkedgeodata.org/sparql`

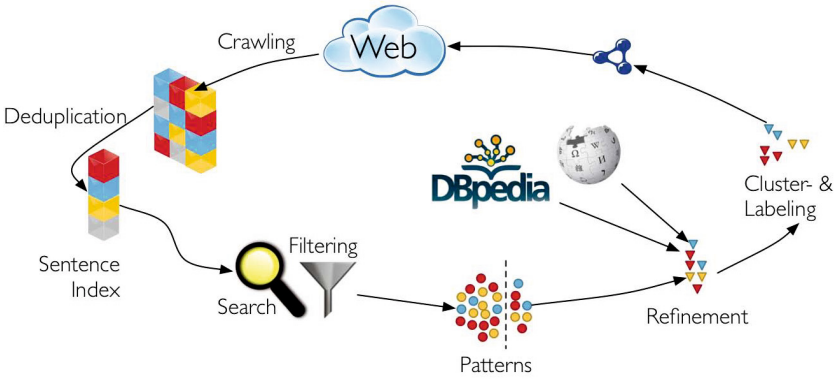


Fig. 1. Overview of the generic time slice-based stream processing

slice duration. For example, this could mean crawling a set of RSS feeds for a duration of 2 hours. We call $D^i_{[t,t+d]}$ a slice of D^i. We will assume that we begin this process at $t = 0$, thus leading to slices $D^i_{[k.d,(k+1).d]}$ with $k \in \mathbb{N}$. The data gathered from all sources during a time slice duration is called a *time slice*. We apply sentence splitting on all slices to generated their elements.

2.2 Deduplication

The aim of the deduplication step is to remove very similar elements from slices before the RDF extraction. This removal accounts for some Web data streams simply repeating the content of one of several other streams. Our deduplication approach is based on measuring the similarity of single elements s_i and s_j found in unstructured streams. Elements of streams are considered to be different iff $qgrams(s_i, s_j) < \theta$, where $\theta \in [0, 1]$ is a similarity threshold and $qgrams(s_i, s_j)$ measures the similarity of two strings by computing the Jaccard similarity of the trigrams they contain. Given that the number of stream items to deduplicate can be very large, we implemented the following two-step approach: For each slice $D^i_{[k.d,(k+1)d]}$, we first deduplicate the elements s^i_j within $D^i_{[k.d,(k+1)d]}$. This results in a duplicate-free data stream $\Delta^i_{[k.d,(k+1)d]} = \{d^i_j : (d^i_j \in D^i_{[k.d,(k+1)d]}) \wedge (\forall s^i_k \in D^i_{[k.d,(k+1)d]}\ \exists d^i_j \in \Delta^i_{[k.d,(k+1)d]}\ qgrams(s^i_k, d^i_j) \geq \theta) \wedge (\forall d^i_j, d^i_k \in \Delta^i_{[k.d,(k+1)d]}\ qgrams(d^i_k, d^i_j) < \theta)\}$. The elements of $\Delta^i_{[k.d,(k+1)d]}$ are then compared to all other elements of the w previous deduplicated streams $\Delta^i_{[(k-1).d,kd]}$ to $\Delta^i_{[(k-w).d,(k-w+1)d]}$, where w is the size of the deduplication window. Only $\Delta^i_{[k.d,(k+1)d]}$ is used for further processing. To ensure the scalability of the deduplication step, we are using deduplication algorithms implemented in the LIMES framework [18]. Table 2 gives an overview of the number of unique data stream items in our dataset when using different deduplication thresholds.

2.3 Pattern Search and Filtering

In order to find patterns we first apply Named Entity Recognition (NER) and Part of Speech (POS) tagging on the deduplicated sentences. RdfLiveNews can use two different ways to extract patterns from annotated text. The POS tag method uses *NNP* and *NNPS*[3] tagged tokens to identify a relation's subject and object, whereas the Named Entity Tag method relies on *Person*, *Location*, *Organization* and *Miscellaneous* tagged tokens. In an intermediate step all consecutive POS and NER tags are merged. An unrefined RdfLiveNews pattern p is now defined as a pair $p = (\theta, \mathcal{S}_\theta)$, where θ is the natural language representation (NLR) of p and $\mathcal{S}_\theta = \{(s_i, o_i) : i \in \mathbb{N}; 1 \leq i \leq n\}$ is the support set of θ, a set of the subject and object pairs. For example the sentence:

David/NNP hired/VBD John/NNP ,/, former/JJ manager/NN of/IN ABC/NNP ./.

would result in the patterns:

$p_1 = (\ [hired], \{(David,\ John)\}$ and
$p_2 = ([,\ former\ manager\ of], \{(John,\ ABC)\})$.

After the initial pattern acquisition step, we filter all patterns to improve their quality. We discarded all patterns that did not match these criteria: The pattern should (1) contain at least a verb or a noun, (2) contain at least one salient word (i.e. a word that is not a stop word), (3) not contain more than one non-alpha-numerical character (except ", ' ‘") and (4) be shorter than 50 characters. Since the resulting list still contains patterns of low quality, we first sort it by the number of elements of the support set $\mathcal{S}_\theta$ and solely select the top 1% for pattern refinement to ensure high quality.

2.4 Pattern Refinement

The goal of this step is to find a suitable *rdfs:range* and *rdfs:domain* as well as to disambiguate the support set of a given pattern. To achieve this goal we first try to find an URI for the subjects and objects in the support set of p by matching the pairs to entries in a knowledge base. With the help of those URIs we can query the knowledge base for the classes (*rdf:type*) of the given resources and compute a common *rdfs:domain* for the subjects of p and *rdfs:range* for the objects respectively. A refined RdfLiveNews pattern p_r is now defined as a quadruple $p_r = (\theta, \mathcal{S}_\theta{}', \delta, \rho)$, where θ is the natural language representation, $\mathcal{S}_\theta{}'$ the disambiguated support set, δ the *rdfs:domain* and ρ the *rdfs:range* of p_r.

To find the URIs of each subject-object pair $(s, o) \in \mathcal{S}_\theta$ we first try to complete the entity name. This step is necessary and beneficial because entities usually get only written once in full per article. For example the newly elected president of the United States of America might be referenced as “President Barack Obama” in the first sentence of a news entry and subsequently be referred to as “Obama”. In order to find the subjects' or objects' full name, we first select all named entities $e \in \mathcal{E}_a$ of the article the pair (s, o) was found in. We then use the

[3] All POS tags can be found in the Penn Treebank Tagset.

longest matching substring between s (or o) and all elements of $\mathcal{E}_a$ as the name of s or o respectively. Additionally we can filter the elements of $\mathcal{E}_a$ to contain only certain NER types. Once the complete names of the entities are found, we can use them to generate a list of URI candidates $\mathcal{C}_{uri}$. This list is generated with the help of a query for the given entity name on a list of surface forms (e.g. "U.S." or "USA" for the *United States of America*), which was compiled by analyzing the *redirect* and *disambiguation* links from Wikipedia as presented in [14]. Each URI candidate $c \in \mathcal{C}_{uri}$ is now evaluated on four different features and the combined score of those features is used to rank the candidates and choose the most probable URI for an entity. The first feature is the *Apriori*-score $a(c)$ of the URI candidate c, which is calculated beforehand for all URIs in the knowledge base by analyzing the number of inbound links of c by the following formula: $a(c) = \log(inbound(c) + 1)$. The second and third features are based on the context information found in the Wikipedia article of c and the news article text (s, o) was found in. For the *global context*-score c_g we apply a co-occurrence analysis of the entities $\mathcal{E}_a$ found in the news article and the entities $\mathcal{E}_w$ found in the Wikipedia article of c. The *global context*-score is now computed as $c_g(\mathcal{E}_a, \mathcal{E}_w) = |\mathcal{E}_a \cap \mathcal{E}_w| \,/\, |\mathcal{E}_a \cup \mathcal{E}_w|$. The *local context*-score c_l is the number of mentions of the second element of the pair (s, o), o in the case of s and vice versa, in $\mathcal{E}_w$. The last feature to determine a URI for an entity is the maximum string similarity *sts* between s (or o) and the elements of the list of surface forms of c. We used the qgram distance[4] as the string similarity metric. We normalize all non-$[0, 1]$ features (c_g, c_l, a) by applying a minimum-maximum normalization of the corresponding scores for $\mathcal{C}_{uri}$ and multiply it with a weight parameter which leads to the overall URI score:

$$c(s, o, uri) = \frac{\frac{\alpha a}{a_{max}} + \frac{\beta c_g}{c_{g_{max}}} + \frac{\gamma c_l}{c_{l_{max}}} + \delta sts}{4}$$

If the URI's score is above a certain threshold $\lambda \in [0, 1]$ we use it as the URI for s, otherwise we create a new URI. Once we have computed the URIs for all pairs $(s, o) \in \mathcal{S}_\theta$ we determine the most likely *domain* and *range* for p_r. This is done by analyzing the *rdf:type* statements returned for each subject or object in $\mathcal{S}_\theta$ from a background knowledge base. Since the DBpedia ontology is designed in such a way, that classes do only have one super-class, we can easily analyze its hierarchy. We implemented two different determination strategies for analyzing the class hierarchy. The first strategy, dubbed "most general", selects the highest class in the hierarchy for each subject (or object) and uses the most occurring class as *domain* or *range* of p_r. The second strategy, dubbed "most specific", works similar to the "most general" strategy with the difference that it uses the most descriptive class to select the *domain* and *range* of p_r.

[4] `http://sourceforge.net/projects/simmetrics/`

2.5 Pattern Similarity and Clustering

In order to cluster patterns according to their meaning, we created a set of similarity measures. A similarity measure takes two patterns p_1 and p_2 as input and outputs the similarity value $s(p_1, p_2) \in [0, 1]$. As a baseline we implemented a qgram measure, which calculates the string similarity between all non stop words of two patterns. Since this baseline measure fails to return a high similarity for semantically related, but not textually similar patterns like "'s attorney ," and "'s lawyer ," we also implemented a Wordnet measure. As a first step the Wordnet similarity measure filters out the stop words of p_1 and p_2 and applies the Stanford lemmatizer on the remaining tokens. Subsequently, for all token combinations of p_1 and p_2, we apply a Wordnet Similarity metric (Path [20], Lin [13] and Wu & Palmer [25]) and select the maximum of all comparisons as the similarity value $s(p_1, p_2)$. As a final similarity measure we created a Wordnet and string similarity measure with the help of a linear combination from the before-mentioned metrics. In this step we also utilize the *domain* and *range* of p_r. If this feature is enabled, a similarity value between two patterns p_1 and p_2 can only be above 0, iff $\{\delta_{p_1}, \rho_{p_1}\} \setminus \{\delta_{p_2}, \rho_{p_2}\} = \emptyset$.

The result of the similarity computation can be regarded as a similarity graph $G = (V, E, \omega)$, where the vertices are patterns and the weight $\omega(p_1, p_2)$ of the edge between two patterns is the similarity of these patterns. Consequently, unsupervised machine learning and in particular graph clustering is a viable way of finding groups of patterns that convey similar meaning. We opted for using the BorderFlow clustering algorithm [19] as it is parameter-free and has already been used successfully in diverse applications including clustering protein-protein interaction data and queries for SPARQL benchmark creation [15]. For each node $v \in V$, the algorithm begins with an initial cluster X containing only v. Then, it expands X iteratively by adding nodes from the direct neighborhood of X to X until X is node-maximal with respect to the border flow ratio described in [15]. The same procedure is repeated over all nodes. As different nodes can lead to the same cluster, identical clusters (i.e., clusters containing exactly the same nodes) that resulted from different nodes are subsequently collapsed to one cluster. The set of collapsed clusters and the mapping between each cluster and the nodes that led to it are returned as result.

2.6 Cluster Labeling and Merging

Based on the clusters $\mathcal{C}$ obtained through the clustering algorithm, this step selects descriptive labels for each cluster $c_i \in \mathcal{C}$, which can afterwards be used to merge the clusters. In the current version, we apply a straightforward majority voting algorithm, i.e. for each cluster c_i, we select the most frequent natural language representation θ (stop words removed) occurring in the patterns of c_i. Finally, we use the representative label of the clusters to merge them using a string similarity and WordNet based similarity measure. This merging procedure can be applied repeatedly to further reduce the number of clusters, but taking into account that those similarity measures are not transitive, we are currently only running it once, as we're more focused on accuracy.

2.7 Mapping to RDF and Publication on the Web of Data

To close the circle of the round-trip pipeline of RdfLiveNews, the following prerequisite steps are required to re-publish the extraction results in a sensible way:

1. The facts and properties contained in the internal data structure of our tool have to be mapped to OWL.
2. Besides the extracted factual information several other aspects and meta data are interesting as well, such as extraction and publication data and provenance links to the text the facts were extracted from.
3. URIs need to be minted to provide the extracted triples as linked data.

Mapping to OWL. Each cluster $c_i \in \mathcal{C}$ represents an *owl:ObjectProperty* $prop_{c_i}$. The *rdfs:domain* and *rdfs:range* of $prop_{c_i}$ is determined by a majority voting algorithm with respect to δ and ρ of all $p_r \in \mathcal{C}$. The *skos:prefLabel*[5] of $prop_{c_i}$ is the label determined by the cluster labeling step and all other NLRs of the patterns in c_i get associated with $prop_{c_i}$ as *skos:altLabels*. For each subject-object pair in $\mathcal{S}_\theta{}'$ we produce a triple by using $prop_{c_i}$ as predicate and by assigning learned entity types from DBpedia or *owl:Thing*.

Provenance Tracking with NIF. Besides converting the extracted facts from the text, we are using the current draft of the NLP Interchange Format (NIF) Core ontology[6] to serialize the following information in RDF: the sentence the triple was extracted from, the extraction date of the triple, the link to the source URL of the data stream item and the publication date of the item on the stream. Furthermore, NIF allows us to link each element of the extracted triple to its origin in the text for further reference and querying.

NIF is an RDF/OWL based format to achieve interoperability between language tools, annotation and resources. NIF offers several URI schemes to create URIs for strings, which can then be used as subjects for annotation. We employ the NIF URI scheme, which is grounded on URI fragment identifiers for text (RFC 5147[7]). NIF was previously used by NERD [21] to link entities to text. For our use case, we extended NIF in two ways: (1) we added the ability to represent extracted triples via the ITS 2.0 / RDF Ontology[8]. *itsrdf:taPropRef* is an *owl:AnnotationProperty* that links the NIF String URI to the *owl:ObjectProperty* by RdfLiveNews. The three links from the NIF String URIs (str_1, str_2, str_3) to the extracted triple (s, p, o) itself make it well traceable and queryable: $str_1 \mapsto s$, $str_2 \mapsto p$, $str_3 \mapsto o$, $s \mapsto p \mapsto o$. An example of NIF RDF serialization is shown in Listing 1. (2) Although [21] already suggested the minting of new URIs, a concrete method for doing so was not yet researched. In RdfLiveNews we use the source URL of the data stream item to re-publish the facts for individual sentences as linked data. We strip the scheme component (http://) of the source URL and percent encode the ultimate part of the path and the query component[9] and add the md5 encoded sentence to produce the following URI:

[5] `http://www.w3.org/2004/02/skos/`
[6] `http://persistence.uni-leipzig.org/nlp2rdf/ontologies/nif-core#`
[7] `http://tools.ietf.org/html/rfc5147`
[8] `http://www.w3.org/2005/11/its/rdf#`
[9] `http://tools.ietf.org/html/rfc3986#section-3`

```
@base <http://rdflivenews.aksw.org/extraction/www.necn.com/07/04/12/
      Scientists-discover-new-subatomic-partic/landing.html%3FblockID
      %3D735470%26feedID%3D4213/8a1e5928f6815c99b9d2ce613cf24198#>.
## prefixes: please use http://prefix.cc, e.g. http://prefix.cc/rlno
## extracted property + result of linking
rlno:directorOf a owl:ObjectProperty ;
  skos:prefLabel "director of" , skos:altLabel ", director of " ;
  owl:equivalentProperty dbp:director .
## extracted facts:
rlnr:Rolf_Heuer a dbo:Person ;
  rdfs:label "Rolf Heuer"@en ;
  rlno:directorOf dbpedia:CERN .
dbpedia:CERN a owl:Thing ;
  rdfs:label "CERN"@en .
## provenance tracking with NIF:
<char=0,10>   itsrdf:taClassRef dbo:Person ;
  itsrdf:taIdentRef rlnr:Rolf_Heuer .
<char=14,18>  itsrdf:taIdentRef dbpedia:CERN .
<char=11,24>  nif:anchorOf      ", director of"^^xsd:string ;
  itsrdf:taPropRef rlno:directorOf .
## detailed NIF output with context, indices and anchorOf
<char=0,> a nif:String, nif:Context, nif:RFC5147String ;
  nif:isString "Rolf Heuer , director of CERN , said the newly
      discovered particle is a boson , but he stopped just shy of
      claiming outright that it is the Higgs boson itself - an
      extremely fine distinction." ;
  nif:sourceUrl <http://www.necn.com/07/04/12/Scientists-discover-
      new-subatomic-partic/landing.html?blockID=735470&feedID=4213>;
## extraction date:
  dcterms:created "2013-05-09T18:27:08+02:00"^^xsd:dateTime .
## publishing date:
<http://www.necn.com/07/04/12/Scientists-discover-new-subatomic-
    partic/landing.html?blockID=735470&feedID=4213>
  dcterms:created "2012-08-15T14:48:47+02:00"^^xsd:dateTime .
<char=0,10> a nif:String, nif:RFC5147String ;
  nif:referenceContext <char=0,>; nif:anchorOf "Rolf Heuer" ;
  nif:beginIndex "0"^^xsd:long ; nif:endIndex "10"^^xsd:long ;
```

Listing 1. Example RDF extraction of RdfLiveNews

```
http://rdflivenews.aksw.org/extraction/ + example.com:8042/over/ +
    urlencode(there?name=ferret) + / + md5(‘sentence‘)
```

Republication of RDF. The extracted triples are hosted on: `http://rdflivenews.aksw.org`. The data for individual sentences is crawlable via the file system of the Apache2 web server. We assume that source URLs only occur once in a stream when the document is published and the files will not be overwritten. Furthermore, the extracted properties and entities are available as linked data at *http://rdflivenews.aksw.org/{ontology|resource}/$name* and they can be queried via SPARQL at `http://rdflivenews.aksw.org/sparql`.

2.8 Linking

The approach described above generates a set of properties with several labels. In our effort to integrate this data source into the Linked Open Data Cloud, we use the deduplication approach proposed in Section 2.2 to link our set of properties to existing knowledge bases (e.g., DBpedia). To achieve this goal, we

consider the set of properties we generated as set of source instances S while the properties of the knowledge base to which we link are considered to be a set of target T. Two properties $s \in S$ and $t \in T$ are linked iff $trigrams(s,t) \geq \theta_p$, where $\theta_p \in [0,1]$ is the property similarity threshold.

3 Evaluation

The aim of our evaluation was to answer four questions. First, we aimed at testing how well RdfLiveNews is able to disambiguate found entities. Our second goal was to determine if the proposed similarity measures can be used to cluster patterns with respect to their semantic similarity. Third, we wanted to evaluate the quality of the RDF extraction and linking. Finally, we wanted to measure if all computational heavy tasks can be applied in real-time, meaning the processing of one iteration takes less time than its compilation.

For this evaluation we used a list of 1457 RSS feeds as compiled in [10]. This list includes all major worldwide newspapers and a wide range of topics, e.g. *World*, *U.S.*, *Business*, *Science* etc. We crawled this list for 76 hours, which resulted in a corpus, dubbed *100%* of 38 time slices of 2 hours and 11.7 million sentences. The average number of sentences per feed entry is approximately 26.5 and there are 3445 articles on average per time slice. Additionally we created two subsets of this corpus by randomly selecting 1% and 10% of the contained sentences. All evaluations were carried out on a MacBook Pro with a quad-core Intel Core i7 (2GHz), a solid state drive and 16 GB of RAM.

3.1 URI Disambiguation

To evaluate the URI disambiguation we created a gold standard manually. We took the 1% corpus, applied deduplication with a window size of 40 (contains all time slices) and a threshold of 1 (identical sentences), which resulted in a set of 69884 unique sentences. On those sentences we performed the pattern extraction with part of speech tagging as well as filtering. In total we found 16886 patterns and selected the Top 1%, which have been found by 1729 entity pairs. For 473 of those entity pairs we manually selected a URI for subject and object. This resulted in an almost equally distributed gold standard with 456 DBpedia and 478 RdfLiveNews URIs. We implemented a hill climbing approach with random initialization to optimize the parameters (see Section 2.4). The precision of our approach is the ratio between correctly found URIs for subject and object to the number of URIs above the threshold λ as shown in Equation 1. The recall, shown in Equation 2, is determined by the ratio between the number of correct subject and object URIs and the total number of subjects and objects in the gold standard. The F_1 measure is determined as usual by: $F_1 = 2 \cdot \frac{P \cdot R}{P+R}$. We optimized our approach for precision since we can compensate a lower recall and could achieve a precision of **85.01%** where the recall is **40.69%** and the resulting F_1 is **55.03%**. The parameters obtained through the hill-climbing search indicate that the *Apriori*-score is the most influential parameter (1.0), followed by *string-similarity* (0.78), *local-context* (0.6), global context (0.45) and

a URI score threshold of 0.61. If we optimize for F_1, we were able to achieve a F_1 measure of **66.49%** with a precision of **67.03%** and a recall of **65.95%**.

For 487 out of the 934 URI in the gold standard no confident enough URI could be found. The most problems occured for DBpedia URIs which could not be determined in 305 cases, in comparison to 182 URIs for newly created resources. Additionally, for 30 resources RdfLiveNews created new URIs where DBpedia URIs should be used and in 0 cases a DBpedia URI was used where a new resource should be created. The reason for those mistakes are tagging errors, erroneous spellings and missing context information. For example Wikipedia has 97 disambiguations for "John Smith" which can not be disambiguated without prior knowledge.

We used AIDA [11] to compare our results with a state-of-the-art NED algorithm. We configured AIDA with the Cocktailparty setup, which defines the recommended configuration options of AIDA. AIDA achieved an accuracy of 0.57, i.e. 57% of the identifiable entities were correctly disambiguated. The corpus described above provides a difficult challenge due to the small disambiguation contexts and is limited to graphs evolving from two named entities per text. AIDA tries to build dense sub-graphs in a greedy manner in order to perform correct disambiguation. This algorithm would profit from a bigger number of entities per text. The drawback is AIDA needs 2 minutes to disambiguate 25 sentences. Overall, AIDA performs well on arbitrary entities.

$$P = \frac{|s_{uri_c}| + |o_{uri_c}|}{|s_{uri}| + |o_{uri}|} \quad (1) \qquad R = \frac{|s_{uri_c}| + |o_{uri_c}|}{2 \cdot |GS|} \quad (2)$$

3.2 Pattern Clustering

To evaluate the similarity generation as well as the clustering algorithm we relied on the measures Sensitivity, Positive Predictive Value (PPV) and Accuracy. We used the adaptation of those measures as presented in [4] to measure the match between a set of pattern mappings[10] from the gold standard and a clustering result. The gold standard was created by clustering the patterns as presented in the previous section manually. This resulted in a list of 25 clusters with more than 1 pattern and 54 clusters with 1 pattern. Since cluster with a size of 1 would skew our evaluation into unjustified good results, we excluded them from this evaluation.

Sensitivity. With respect to the clustering gold standard, we define sensitivity as the fraction of patterns of pattern mapping i which are found in cluster j. In $Sn_{i,j} = T_{i,j}/N_i$, N_i is the number of patterns belonging to pattern mapping i. We also calculate a pattern mapping-wise sensitivity Sn_{pm_i} as the maximal fraction of patterns of pattern mapping i assigned to the same cluster. $Sn_{pm_i} = max_{j=1}^{m} Sn_{i,j}$ reflects the coverage of pattern mapping i by its best-matching cluster. To characterize the general sensitivity of a clustering result, we compute

[10] A pattern mapping maps NLRs to RDF properties.

a clustering-wise sensitivity as the weighted average of Sn_{pm_i} over all pattern mappings: $Sn = \frac{\sum_{i=1}^{n} N_i Sn_{pm_i}}{\sum_{i=1}^{n} N_i}$.

Positive Predictive Value. The positive predictive value is the proportion of members of cluster j which belong to pattern mapping i, relative to the total number of members of this cluster assigned to all pattern mappings. $PPV_{i,j} = T_{i,j} / \sum_{i=1}^{n} T_{i,j} = T_{i,j}/T_{.j}$

$T_{.j}$ is the sum of column j. We also calculate a cluster-wise positive predictive value PPV_{cl_j}, which represents the maximal fraction of patterns of cluster j found in the same annotated pattern mapping. $PPV_{cl_j} = max_{i=1}^{n} PPV_{i,j}$ reflects the reliability with which cluster j predicts that a pattern belongs to its best-matching pattern mapping. To characterize the general PPV of a clustering result as a whole, we compute a clustering-wise PPV as the weighted average of PPV_{cl_j} over all clusters: $PPV = \frac{\sum_{j=1}^{m} T_{.j} PPV_{cl_j}}{\sum_{j=1}^{m} T_{.j}}$.

Accuracy. The geometric accuracy (Acc) indicates the tradeoff between sensitivity and positive predictive value. It is obtained by computing the geometrical mean of the Sn and the PPV: $Acc = \sqrt{Sn \cdot PPV}$.

We evaluated the three similarity measures with respect to the underlying WordNet similarity metric (see Section 2.5). Furthermore we varied the clustering similarity threshold between 0.1 and 1 with a 0.1 step size. In case of the qgram and WordNet similarity metric we performed a grid search on the WordNet and qgram parameter in $[0, 1]$ with a step size of 0.05. We achieved the best configuration with the qgram and WordNet similarity metric with an accuracy of **82.45%**, a sensitivity of **71.17%** and a positive predictive value of **95.51%**. The best WordNet metric is Lin, the clustering threshold 0.3 and the qgram parameter is with 0.45 significantly less influential than the WordNet parameter with 0.75. As a reference value, the plain WordNet similarity metric achieved an accuracy of 78.86% and the qgram similarity metric an accuracy of 69.1% in their best configuration.

3.3 RDF Extraction and Linking

To assess the quality of the RDF data extracted by RdfLiveNews, we sampled the output of our approach and evaluated it manually. We generated five different evaluation sets. Each set may only contain triples with properties of clusters having at least $i = 1 \ldots 5$ patterns. We selected 100 triples (if available) randomly for each test set. As the results in Table 1 show, we achieve high accuracy on subject and object disambiguation. As expected, the precision of our approach grows with the threshold for the minimal size of clusters. This is simply due to the smaller clusters having a higher probability of containing outliers and thus noise.

The results of the linking with DBpedia (see Table 3) showed the mismatch between the relations that occur in news and the relations designed to model encyclopedic knowledge. While some relations such as `dbo:director` are used commonly in news streams and in the Linked Data Cloud, relations with a more volatile character such as `rlno:attorney` which appear frequently in news text are not mentioned in DBpedia.

Table 1. Accuracy of RDF Extraction for subject (S), predicates (P) and objects (O) on 1% dataset with varying cluster sizes E_i

E_i	1	2	3	4	5
S_{Acc}	0.81	0.88	0.86	0.857	0.804
P_{Acc}	0.86	0.89	0.90	0.935	1.00
O_{Acc}	0.93	0.91	0.90	0.948	0.941
$Total_{Acc}$	0.86	0.892	0.885	0.911	0.906
$\|E_i\|$	100	100	100	77	51
$\|P\| \in \|E_i\|$	28	22	12	6	1

Table 2. Number of non-duplicate sentences in 1% of the data extracted from 1457 RSS feeds within a window of 10 time slices (2h each). The second column shows the original number of sentences without duplicate removal.

Time Slice	No deduplication	$\theta = 1.0$	$\theta = 0.95$	$\theta = 0.9$
1	2997	2764	2764	2759
5	3047	2335	2334	2327
10	3113	2033	2040	2022
15	2927	1873	1868	1866
20	3134	1967	1966	1949
25	3065	1936	1932	1924
30	3046	1941	1940	1933

Table 3. Example for linking between RdfLiveNews and DBpedia

RdfLiveNews-URI	DBpedia-URI	Sample of cluster
rlno:directorOf	dbo:director	[manager], [, director of], [, the director of]
rlno:spokesperson	dbo:spokesperson	[, a spokeswoman for], [spokesperson], [, a spokesman for]
rlno:attorney	—	['s attorney ,], ['s lawyer ,], [attorney]

3.4 Scalability

In order to perform real-time RDF extraction, the processing of the proposed pipeline needs to be done in less time than its acquisition requires. This also needs to be true for a growing list of RSS feeds. Therefore, we analyzed the time each module needed in each iteration and compared these values between the three test corpora. An early approximation of this evaluation implied that the pipeline indeed was not fast enough, which led to the parallelization of the pattern refinement and similarity generation. The results of this evaluation can be seen in Figure 2. With an average time slice processing time of about 20 minutes for the 100% corpus (2.2 minutes for 10% and 30s for 1%), our approach is clearly fit to handle up to 1500 RSS and more. The spike in the first iteration results out of the fact that RSS feeds contain the last n previous entries, which leads to a disproportional large first time slice. The most time consuming modules are the deduplication, tagging and cluster merging. To tackle these bottlenecks we can for example parallelize sentence tagging and the deduplication.

The results of the growth evaluation for patterns until iteration 30 can be seen in Figure 3. The number of patterns grows with the factor of 3 from 1% to 10% and 10% to 100% corpora. Also, the number of patterns found by more than one subject-object pair increases approximately by factor 2. Additionally we observed a linear growth for all patterns (also for patterns with $|\mathcal{S}'_\theta| > 1$) and 100% showing the highest growth rate with a factor 2.5 over 10% and 4.8 over 10%.

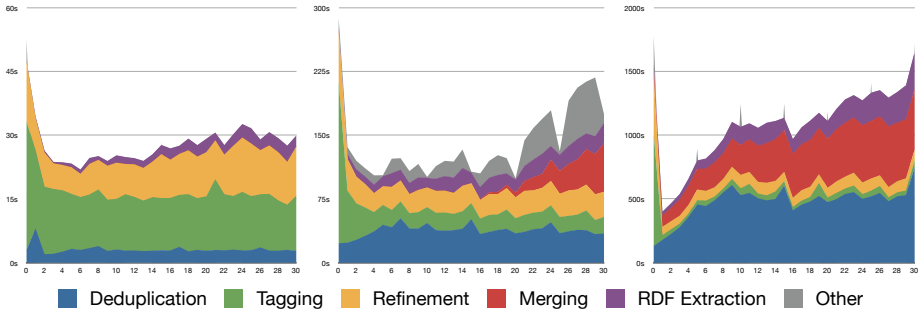


Fig. 2. Runtimes for different components and corpora (1% left, 10% middle, 100% right) per iteration

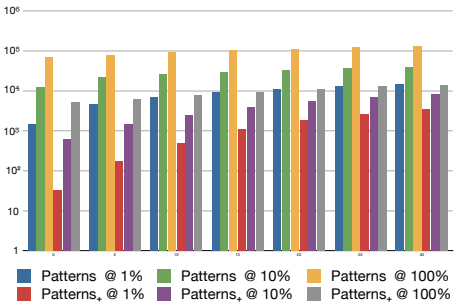


Fig. 3. Number of patterns (log scale) and patterns with $|\mathcal{S}'_\theta| > 1$ (Patterns$_+$) for iterations and test corpus

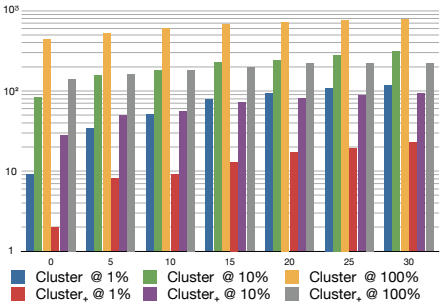


Fig. 4. Number of clusters (log scale) and clusters with $|\mathcal{C}| > 1$ (Cluster$_+$) for iterations and test corpus

The results of the growth evaluation for clusters can be seen in Figure 4. The evaluation shows that the number of clusters increases by a factor of 2.5 from 1% to 10% and 10% to 100%. Moreover, approximately 25% of all cluster have more than 1 pattern and the number of clusters grows linear for 1% and 10% but for the 100% corpus it seems to coverage to 800. The same holds true for clusters with more then one pattern, as they stop to grow at around 225 clusters.

4 Related Work

While Semantic Web applications rely on formal, machine understandable languages such as RDF and OWL, enabling powerful features such as reasoning and expressive querying, humans use Natural Language (NL) to express semantics. This gap between the two different languages has been filled by Information Extraction (IE) approaches, developed by the Natural Language Processing (NLP) research community [23], whose goal is to find desired pieces of information, such as concepts (hierarchy of terms which are used to point to shared definitions), entities (name, numeric expression, date) and facts in natural language texts and

print them in a form that is suitable for automatic querying and processing. Ever since the advent of the Linked Open Data initiative[11], IE is also an important key enabler for the Semantic Web. For example, LODifier ([2], [6]) combines deep semantic analysis with named entity recognition, word-sense disambiguation and controlled Semantic Web vocabularies. FOX [17] uses ensemble learning to improve the F-score of IE tools. The BOA framework [9] uses structured data as background knowledge for the extraction of natural language patterns, which are subsequently employed to extract additional RDF data from natural language text. The authors of [16] propose a simple model for fact extraction in real-time taking into account the difficult challenges that timely fact extraction on frequently updated data entails. A specific application for the news domain is described in [24], wherein a knowledge base of entities for the French news agency AFP is populated.

State-of-the-art open-IE systems such as ReVerb automatically identify and extract relationships from text, relying on (in the case of ReVerb) simple syntactic constraints expressed by verbs [7]. The authors of [5] present a novel pattern clusters method for nominal relationship classification using an unsupervised learning environment, which makes the system domain and language-independent. [22] shows how lexical patterns and semantic relationships can be learned from concepts in Wikipedia.

5 Conclusion and Future Work

In this paper, we presented RdfLiveNews, a framework for the extraction of RDF from unstructured data streams. We presented the components of the RdfLiveNews framework and evaluated its disambiguation, clustering, linking and scalability capabilities as well as its extraction quality. We are able to disambiguate resources with a precision of 85%, cluster patterns with an accuracy of 82.5% and extract RDF with an total accuracy of around 90% and handle two hour time slices with around 300.000 sentences within 20 min on a small server. In future work, we will extend our approach to also cover datatype properties. For example from the sentence "..., Google said Motorola Mobility contributed revenue of US$ 1.25 billion for the second quarter." the triple *dbpedia:Google rlno:says "Motorola Mobility contributed revenue of US$ 1.25 billion for the second quarter"* can be extracted. Additionally we plan to integrate DeFacto [12], which is able to verify or falsify a triple extracted by RdfLiveNews. Finally, we will extend our approach with temporal logics to explicate the temporal scope of the triples included in our knowledge base.

References

1. Auer, S., Lehmann, J., Ngonga Ngomo, A.-C., Zaveri, A.: Introduction to linked data and its lifecycle on the web. In: Rudolph, S., Gottlob, G., Horrocks, I., van Harmelen, F. (eds.) Reasoning Weg 2013. LNCS, vol. 8067, pp. 1–90. Springer, Heidelberg (2013)

[11] `http://linkeddata.org/`

2. Augenstein, I., Padó, S., Rudolph, S.: Lodifier: Generating linked data from unstructured text. In: Simperl, E., Cimiano, P., Polleres, A., Corcho, O., Presutti, V. (eds.) ESWC 2012. LNCS, vol. 7295, pp. 210–224. Springer, Heidelberg (2012)
3. Berners-Lee, T., Hendler, J., Lassila, O.: The semantic web. Scientific American 284(5), 34–43 (2001)
4. Brohée, S., van Helden, J.: Evaluation of clustering algorithms for protein-protein interaction networks. BMC Bioinformatics (2006)
5. Davidov, D., Rappoport, A.: Classification of semantic relationships between nominals using pattern clusters. ACL (2008)
6. Exner, P., Nugues, P.: Entity extraction: From unstructured text to dbpedia rdf triples. In: Rizzo, G., Mendes, P., Charton, E., Hellmann, S., Kalyanpur, A. (eds.) Web of Linked Entities Workshop (WoLE 2012) (2012)
7. Fader, A., Soderland, S., Etzioni, O.: Identifying relations for open information extraction. In: EMNLP, pp. 1535–1545. ACL (2011)
8. Gaag, A., Kohn, A., Lindemann, U.: Function-based solution retrieval and semantic search in mechanical engineering. In: IDEC 2009, pp. 147–158 (2009)
9. Gerber, D., Ngonga Ngomo, A.-C.: Bootstrapping the linked data web. In: 1st Workshop on Web Scale Knowledge Extraction @ ISWC 2011 (2011)
10. Goldhahn, D., Eckart, T., Quasthoff, U.: Building large monolingual dictionaries at the leipzig corpora collection: From 100 to 200 languages. In: LREC (2012)
11. Hoffart, J., Yosef, M.A., Bordino, I., Fürstenau, H., Pinkal, M., Spaniol, M., Taneva, B., Thater, S., Wiegand, M., Weikum, G.: Robust disambiguation of named entities in text. In: Conference on Empirical Methods in Natural Language Processing: EMNLP 2011, Proceedings of the Conference, Edinburgh, United Kingdom, Stroudsburg, PA, July 27-31, pp. 782–792. ACL, MP (2011) 978-1-937284-11-4
12. Lehmann, J., Gerber, D., Morsey, M., Ngonga Ngomo, A.-C.: DeFacto - Deep Fact Validation. In: Cudré-Mauroux, P., Heflin, J., Sirin, E., Tudorache, T., Euzenat, J., Hauswirth, M., Parreira, J.X., Hendler, J., Schreiber, G., Bernstein, A., Blomqvist, E. (eds.) ISWC 2012, Part I. LNCS, vol. 7649, pp. 312–327. Springer, Heidelberg (2012)
13. Lin, D.: An Information-Theoretic Definition of Similarity. In: Shavlik, J.W., Shavlik, J.W. (eds.) ICML, pp. 296–304. Morgan Kaufmann (1998)
14. Mendes, P.N., Jakob, M., Garcia-Silva, A., Bizer, C.: DBpedia Spotlight: Shedding Light on the Web of Documents. In: I-SEMANTICS. ACM International Conference Proceeding Series, pp. 1–8. ACM (2011)
15. Morsey, M., Lehmann, J., Auer, S., Ngonga Ngomo, A.-C.: Dbpedia sparql benchmark - performance assessment with real queries on real data. In: Aroyo, L., Welty, C., Alani, H., Taylor, J., Bernstein, A., Kagal, L., Noy, N., Blomqvist, E. (eds.) ISWC 2011, Part I. LNCS, vol. 7031, pp. 454–469. Springer, Heidelberg (2011)
16. Nakashole, N., Weikum, G.: Real-time population of knowledge bases: opportunities and challenges. In: Proceedings of AKBC-WEKEX (2012)
17. Ngonga Ngomo, A.-C., Heino, N., Lyko, K., Speck, R., Kaltenböck, M.: SCMS - Semantifying Content Management Systems. In: Aroyo, L., Welty, C., Alani, H., Taylor, J., Bernstein, A., Kagal, L., Noy, N., Blomqvist, E. (eds.) ISWC 2011, Part II. LNCS, vol. 7032, pp. 189–204. Springer, Heidelberg (2011)
18. Ngonga Ngomo, A.-C.: On link discovery using a hybrid approach. J. Data Semantics 1(4), 203–217 (2012)
19. Ngonga Ngomo, A.-C., Schumacher, F.: Borderflow: A local graph clustering algorithm for natural language processing. In: Gelbukh, A. (ed.) CICLing 2009. LNCS, vol. 5449, pp. 547–558. Springer, Heidelberg (2009)

20. Pedersen, T., Patwardhan, S., Michelizzi, J.: Wordnet: Similarity - measuring the relatedness of concepts. In: AAAI (2004)
21. Rizzo, G., Troncy, R., Hellmann, S., Brümmer, M.: NERD meets NIF: Lifting NLP extraction results to the linked data cloud. In: LDOW, France (2012)
22. Ruiz-Casado, M., Alfonseca, E., Castells, P.: Automatising the learning of lexical patterns: An application to the enrichment of wordnet by extracting semantic relationships from wikipedia (2007)
23. Sarawagi, S.: Information extraction. Found. Trends Databases (2008)
24. Stern, R., Sagot, B.: Population of a knowledge base for news metadata from unstructured text and web data. In: Proceedings of the AKBC-WEKEX (2012)
25. Wu, Z., Palmer, M.S.: Verb semantics and lexical selection. In: Pustejovsky, J. (ed.) ACL, pp. 133–138. Morgan Kaufmann Publishers / ACL (1994)

One License to Compose Them All

A Deontic Logic Approach to Data Licensing on the Web of Data

Guido Governatori[1,⋆], Antonino Rotolo[2], Serena Villata[3,⋆⋆], and Fabien Gandon[3]

[1] NICTA Queensland Research Laboratory
[2] University of Bologna
[3] INRIA Sophia Antipolis

Abstract. In the domain of Linked Open Data a need is emerging for developing automated frameworks able to generate the licensing terms associated to data coming from heterogeneous distributed sources. This paper proposes and evaluates a deontic logic semantics which allows us to define the deontic components of the licenses, i.e., permissions, obligations, and prohibitions, and generate a composite license compliant with the licensing items of the composed different licenses. Some heuristics are proposed to support the data publisher in choosing the licenses composition strategy which better suits her needs w.r.t. the data she is publishing.

1 Introduction

Following the Open Data movement[1], several data hubs are being created by public bodies from single cities through to supra national organizations like the European Union with the final aim to improve the transparency and efficiency of such public bodies and organizations. In this context, the data is openly published on the Web using different data models (e.g., RDF, schema.org, CSV). However, even if this movement is receiving more attention in the last years, still much more effort is required to publish open data on the Web, possibly in a machine-readable format in such a way that data could be interlinked, supporting the growth of the Web of Data [3,13]. One open problem in this context is *quality assessment* with a particular attention to provenance information [12]. More precisely, part of the self-description of the data consists in the licensing terms which specify the admitted use and re-use of the data by third parties. This issue is relevant both for *i)* Linked Data publication as underlined in the "7 Best Practices for Producing Linked Data"[2] where it is required to specify an appropriate license for the data, and *ii)* Open Data publication since the possibility to express constraints on the reuse of the data would encourage the publication of more open data. In this paper, we answer the research question: *How to express the licensing terms associated to*

⋆ NICTA is funded by the Australian Government as represented by the Department of Broadband, Communications and the Digital Economy and the Australian Research Council through the ICT Centre of Excellence program.

⋆⋆ The author acknowledges support of the DataLift Project ANR-10-CORD-09 founded by the French National Research Agency.

[1] `http://www.w3.org/TR/gov-data/`

[2] `http://www.w3.org/2011/gld/wiki/Linked_Data_Cookbook`

H. Alani et al. (Eds.): ISWC 2013, Part I, LNCS 8218, pp. 151–166, 2013.

data coming from heterogeneous distributed sources? This research question could be answered by linking the datasets to normative documents describing the licenses under which they are released. However, this solution is far from the Web of Data philosophy where the meta-information about the datasets should be expressed both in a human and a machine-readable format to allow further reasoning steps. Thus the research question breaks down into the following subquestions: *i)* How to express the deontic component of the licensing terms in a machine-readable format?, and *ii)* How to compose in a compliant and automated way the licensing terms associated to a set of heterogeneous data to produce a single composite license?

First, we introduce a lightweight vocabulary called `l4lod`[3] (Licenses for Linked Open Data) which is composed by the three main deontic components, i.e., Obligations, Permissions and Prohibitions, and provides an alignment with the other licenses vocabularies. It is used to express the machine-readable composite license.

Second, we rely on the deontic logic paradigm [24] to address the problem of *reconciling* a set of licenses associated to heterogeneous datasets whose information items are returned together for consumption, e.g., resulting from a single SPARQL query over distributed datasets released under different licenses. Assuming that these datasets provide the consumer with their own licensing terms, we propose and evaluate a deontic logic semantics which automatically returns to the consumer a so called *composite license* which is compliant with the normative semantics of each single license composing it.

The rationale of this work is to support both consumers and publishers of Linked Open Data. On the one hand, as a consumer it is fundamental to know the kind of operations you are permitted to perform on the data to avoid data misuses. On the other hand, we support the publisher to decide which heuristics better suits her needs about the composition of the licenses associated to her data, e.g., the composite license with less obligations and more permissions is preferred to the others.

The reminder of the paper is as follows. Section 2 starts with an analysis of the Linked Data cloud from the licenses point of view and presents the `l4lod` vocabulary. In Section 3, we present our deontic logic to represent and reason over the licensing terms together with the heuristics to guide licenses composition. In Section 4, we evaluate our approach using the SPINdle logic reasoner. In Section 5 we present the existing research, and we compare it with the proposed approach.

2 Licenses for Linked Open Data

The first issue to be addressed with respect to the use of licenses in Linked Open Data (LOD) is to understand how many datasets of the LOD cloud[4] are actually licensed, and, at a later stage, which are the more popular licenses adopted in those datasets. In order to perform such analysis, we crawled the LOD cloud[5] with a total of 235 datasets considered. The results of this analysis are as follows:

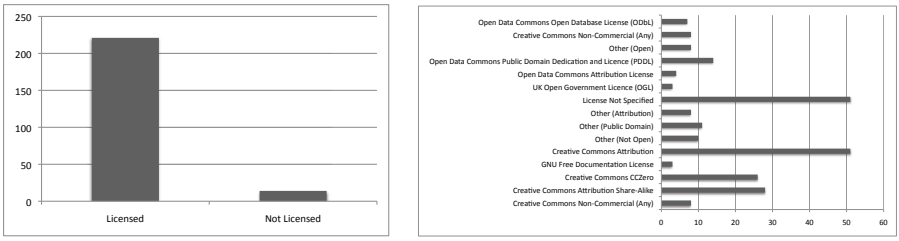
[3] `http://ns.inria.fr/l4lod/`

[4] `http://lod-cloud.net/`

[5] The Data Hub: Linking Open Data Cloud. Retrieved May 02, 2013 (UTC). `http://datahub.io/group/lodcloud`

Licensed-Not Licensed. 221 datasets out of 235 are licensed in some way (Figure 1(a)). The licensing terms are often reported in the VOID meta-data[6] using the `dcterms:license` or the `dcterms:rights` properties of Dublin Core[7]. The license is not usually explicited in a machine-readable format, i.e., the URI of the license is given, but it brings to the human-readable version only (around 95%).

Licenses distribution. The most adopted license is Creative Commons Attributions (CC-BY)[8] (51 out of 221 datasets), and the Creative Commons (CC) licenses [1] in general represent the 51% of the licenses used on the LOD cloud (Figure 1(b)). Other popular licenses are Open Data Commons (ODC) ones[9] [18] (11%), and other licenses from specific institutions (18%)[10].



(a) (b)

Fig. 1. Surveying the Linked Data Cloud: licensed data statistics

We introduce the `l4lod` lightweight vocabulary (Licenses for Linked Open Data) which is used to collect and align existing vocabularies which specify with different granularity levels the licensing terms associated to the data. `l4lod` is adopted in our framework as reference vocabulary to specify in a machine-readable format the licensing terms associated to the composite license we automatically generate. Moreover, starting by the observation that not all licensed works are *creative works* [13], `l4lod` may be used to specify the deontic components for those licenses outside CC, like for instance ODC licenses, and the Open Government License (OGL) [11], as through the Open Digital Rights Language (ODRL) vocabulary[12]. The fine grained specification of licensing terms in a machine-readable format is the goal of the ODRL vocabulary, while the aim of `l4lod` is to describe the composite license at the level of its basic deontic components. We are currently investigating how to use the ODRL vocabulary to address the `l4lod` requirements as an ODRL Profile.

[6] `http://www.w3.org/TR/void/`
[7] `http://purl.org/dc/terms/`
[8] `http://creativecommons.org/licenses/by/3.0/`
[9] `http://opendatacommons.org/licenses/`
[10] LOD cloud highlighting licenses distribution available at `http://ns.inria.fr/l4lod/`
[11] `http://www.nationalarchives.gov.uk/doc/open-government-licence/`
[12] `http://www.w3.org/community/odrl/two/model/`

We define the class `License` which is equivalent to `cc:License`[13] and `limo:LicenseModel`[14], and three basic deontic properties which are respectively `permits`, `prohibits`, and `obliges`. These properties connect each license with its own elements: `Reproduction, Derivative, Distribution, Sharing, Using, CommercialExpl, Publishing` (for *Permissions*), `Attribution, ShareAlike, Citation` (for *Obligations*), and `NoCommercial, NoDerivative` (for *Prohibitions*). The vocabulary does not provide an exhaustive set of properties for licenses definition. Implementations are free to extend `l4lod` to add further elements. In the vocabulary, we distinguish between facts (i.e., rights as in the class `License`) and their representation. That is why we introduce the `licensingTerms` property to connect the license to its human-readable counterpart (domain `dc:LicenseDocument`). Further distinctions, e.g., among facts/information, collections of facts, are out of the scope of this vocabulary and they are carried out by other vocabularies (e.g., ODRL).

The vocabulary considers, among others, the alignment with the following vocabularies: the CC vocabulary, the ODRL vocabulary[15], the LiMo vocabulary, the Dublin Core vocabulary, the Waiver vocabulary[16], the Description of a Project vocabulary (`doap`)[17], the Ontology Metadata vocabulary (`omv`)[18], the Data Dictionary for Preservation Metadata (`premis`)[19], the Vocabulary Of Attribution and Governance (`voag`)[20].

3 Defeasible Deontic Logic for Licenses Composition

We propose an extension of Defeasible Logic, revising earlier works [8,9], to handle license composition. Dealing with this issue requires reasoning about two components:

Factual and ontology component: the first component is meant to describe the facts with respect to which Web of Data licenses are applied as well as the ontology of concepts involved by licenses (thus modeling, e.g., concept inclusion);

Deontic component: the second component aims at capturing the deontic aspects of Web of Data licenses, thus offering mechanisms for reasoning about obligations, prohibitions, and permissions in force in each license, and in their composition.

In this paper, we basically focus on the deontic component, even though, for the sake of completeness, we illustrate the proposed method by also handling, in standard Defeasible Logic, the factual and ontology component, as done in [4]. However, standard Defeasible Logic is just an option, and the factual and ontology component can be handled in any other suitable logic and by resorting to a separate reasoner. Also, notice that we assume that all licenses share a same ontology, or the ontologies are aligned.

The formal language of the logic is rule-based. Literals can be plain, such as $p, q, r \ldots$, or modal, such $\mathrm{O}p$ (obligatory), $\mathrm{P}p$ (permitted), and $\mathrm{F}p$ (forbidden/prohibited). Ontology

[13] `http://creativecommons.org/ns`
[14] `http://purl.org/LiMo/0.1`
[15] `http://w3.org/ns/odrl/2/`
[16] `http://vocab.org/waiver/terms/.html`
[17] `http://usefulinc.com/ns/doap`
[18] `http://omv2.sourceforge.net/index.html`
[19] `http://bit.ly/premisOntology`
[20] `http://voag.linkedmodel.org/schema/voag`

rules work as regular Defeasible Logic rules for deriving plain literals, while the logic of deontic rules provide a constructive account of the basic deontic modalities (obligation, prohibition, and permission). However, while we assume that all licenses share a same ontology, the purpose of the formalism is mainly to establish the conditions to derive *different* deontic conclusions from *different* licenses, and check whether they are compatible so that they can be attributed to a composite license. Hence, we need to keep track of how these deontic conclusions are obtained. To this purpose, deontic rules (and, as we will see, their conclusions) are parametrized by labels referring to licenses.

An ontology rule such as $a_1,\ldots,a_n \Rightarrow b$ supports the conclusion of b, given $a_1,\ldots,a_n$, and so it states that, from the viewpoint of any license any instance enjoying $a_1,\ldots,a_n$ is also an instance of b. On the contrary, rules as $a, \mathrm{O}b \Rightarrow^{l_2}_{\mathrm{O}} p$ state that, if a is the case and b is obligatory, then $\mathrm{O}p$ holds in the perspective of license l_2, i.e., p is obligatory for l_2.

The proof theory we propose aims at offering an efficient method for reasoning about the deontic component of each license and, given that method, for combining different licenses, checking their compatibility, and establishing what deontic conclusions can be drawn from the composite license. In other words, if $l_c = l_1 \odot \cdots \odot l_n$ is the composite license obtained from $l_1,\ldots,l_n$, the conclusions derived in the logic for $l_1,\ldots,l_n$ are also used to establish those that hold in l_c.

The reader may argue about the choice of defeasible deontic logics. A simpler approach would be to foster the adoption of standardized licenses and assign them a URI. Then, a basic URI comparison can trigger the allowed/appropriate usages of the data. However, even if we support such kind of standardization, we believe that it is far from the present situation where different licenses are used on the Web, from the basic purpose licenses up to the national ones. Dealing with licenses composition requires reasoning about all deontic provisions, handling and solving normative conflicts arising from deontically incompatible licenses, and exceptions. A few formalisms can do that. Defeasible deontic logic is one of the best candidates, as all aspects are managed in an efficient and computationally tractable way.

3.1 Formal Language and Basic Concepts

The basic language is defined as follows. Let $\mathrm{Lic} = \{l_1, l_2, \ldots, l_n\}$ be a finite set of licenses. Given a set PROP of *propositional atoms*, the set of *literals* Lit is the set of such atoms and their negation; as a convention, if q is a literal, $\sim q$ denotes the complementary literal (if q is a positive literal p then $\sim q$ is $\neg p$; and if q is $\neg p$, then $\sim q$ is p). Let us denote with $\mathrm{MOD} = \{\mathrm{O},\mathrm{P},\mathrm{F}\}$ the set of basic deontic modalities. The set ModLit of modal literals is defined as follows: i) if $X \in \mathrm{MOD}$ and $l \in \mathrm{Lit}$, then Xl and $\neg Xl$ are modal literals, ii) nothing else is a modal literal.

Let Lbl be a set of arbitrary labels. Every rule is of the type $r : A(r) \hookrightarrow^{x}_{Y} C(r)$, where

1. $r \in \mathrm{Lbl}$ is the name of the rule;
2. $A(r) = \{a_1,\ldots,a_n\}$, the *antecedent* (or *body*) of the rule, is a finite set denoting the premises of the rule. If r is an ontology rule, then each a_i, $1 \leq i \leq n$, belongs to Lit, otherwise it belongs to $\mathrm{Lit} \cup \mathrm{ModLit}$;
3. $\hookrightarrow \in \{\rightarrow, \Rightarrow, \rightsquigarrow\}$ denotes the type of the rule;

4. if r is a deontic rule, $Y = \mathrm{O}$ represents the type of conclusion obtained[21]; otherwise (for ontology rules), $Y \in \emptyset$;
5. if r is a deontic rule, $x \in \mathrm{Lic}$ indicates to which license the rule refers to; otherwise (for ontology rules), $x \in \emptyset$;
6. $C(r) = b \in \mathrm{Lit}$ is the *consequent* (or *head*) of the rule.

The intuition behind the different arrows is the following. *Strict rules* have the form $a_1, \ldots, a_n \rightarrow^x_Y b$. *Defeasible rules* have the form $a_1, \ldots, a_n \Rightarrow^x_Y b$. A rule of the form $a_1, \ldots, a_n \rightsquigarrow^x_Y b$ is a *defeater*. Analogously, for ontology rules, where arrows do not have superscripts and subscripts. The three types of rules establish the strength of the relationship. Strict rules provide the strongest connection between a set of premises and their conclusion: whenever the premises are deemed as indisputable so is the conclusion. Defeasible rules allow to derive the conclusion unless there is evidence for its contrary. Finally, defeaters suggest that there is a connection between its premises and the conclusion not strong enough to warrant the conclusion on its own, but such that it can be used to defeat rules for the opposite conclusion.

A multi-license theory is the knowledge base which is used to reason about the applicability of license rules under consideration.

Definition 1. *A* multi-license theory *is a structure* $D = (F, L, R^c, \{R^{\mathrm{O}^l}\}_{l \in \mathrm{Lic}}, \succ)$*, where*

- $F \subseteq \mathrm{Lit} \cup \mathrm{ModLit}$ *is a finite set of facts;*
- $L \subseteq \mathrm{Lic}$ *is a finite set of licenses;*
- R^c *is a finite set of ontology rules;*
- $\{R^{\mathrm{O}^l}\}_{l \in \mathrm{Lic}}$ *is finite family of sets of obligation rules;*
- $\succ$ *is an acyclic relation (called superiority relation) defined over* $(R^c \times R^c) \cup (R^{\mathrm{O}^l} \times R^{\mathrm{O}^{l'}})$*, where* $R^{\mathrm{O}^l}, R^{\mathrm{O}^{l'}} \in \{R^{\mathrm{O}^l}\}_{l \in \mathrm{Lic}}$[22].

$R[b]$ and $R^X[b]$ with $X \in \{c, \mathrm{O}^l | l \in \mathrm{Lic}\}$ denote the set of all rules whose consequent is b and of all rules (of type X). Given a set of rules R the sets R_{s}, R_{sd}, and R_{dft} denote, respectively, the subsets of R of strict rules, defeasible rules, and defeaters.

3.2 Proof Theory

A *proof* P of length n is a finite sequence $P(1), \ldots, P(n)$ of tagged literals of the type $+\Delta^X q$, $-\Delta^X q$, $+\partial^X q$ and $-\partial^X q$, where $X \in \{c, Y^l | l \in \mathrm{Lic}, Y \in \mathrm{MOD}\}$. The proof conditions below define the logical meaning of such tagged literals. As a conventional notation, $P(1...i)$ denotes the initial part of the sequence P of length i. Given a multi-license theory D, $+\Delta^X q$ means that literal q is provable in D with the mode X using only facts and strict rules, $-\Delta^X q$ that it has been proved in D that q is not definitely provable in D with the mode X, $+\partial^X q$ that q is defeasibly provable in D with the mode X, and $-\partial^X q$ that it has been proved in D that q is not defeasibly provable in D with the mode X[23].

[21] We will see why we do not need rules for prohibitions and permissions.

[22] Notice that we may have that $l = l'$.

[23] As we will see, we shall adopt a reading of permissions according to which they can only be defeasible. Hence, we will not define the cases $\pm\Delta^{Y^l} q$ where $Y = \mathrm{P}$.

Given $\# \in \{\Delta, \partial\}$, $P = P(1), \ldots, P(n)$ *is a proof for* p in D for the license l iff $P(n) = +\#^l p$ when $p \in$ Lit, $P(n) = +\#^{X^l} q$ when $p = Xq \in$ ModLit, and $P(n) = -\#^{Y^l} q$ when $p = \neg Yq \in$ ModLit.

The proof conditions aim at determining what conclusions can be obtained within composite licenses by using the source licenses. Three heuristics have been proposed for this purpose [6,23]:

- **OR-composition:** if at least one of the licenses owns a clause then also l_c owns it;
- **AND-composition:** if all the licenses own a clause then also l_c owns it;
- **Constraining-value:** the most constraining clause among those offered by the single licenses is included in l_c.

In this paper, we concentrate on deontic effects of licenses, thus working on the obligations, prohibitions, permissions entailed by the composition of a given set of licenses (instead of the composition of the clauses). Also, since the constraining-value heuristics requires to fully model the idea of concept inclusion (thus working also on the ontology part; see discussion in [21]), here we focus the first two heuristics, reframed as:

- **OR-composition:** l_c entails a deontic effect if there is at least one license that entails such effect (and no license prevents it).
- **AND-composition:** l_c entails a deontic effect if all licenses entail it.

In the next sections, we will show by means of examples, how the AND- and OR-heuristics operate in the logic, including the derived conclusions.

Some notational conventions and concepts that we will use throughout the remainder of this section: *i)* let $l_c = l_1 \odot \cdots \odot l_n$ be any composite license that can be obtained from the set of licenses $L_c = \{l_1, \ldots, l_n\} \subseteq L$; *ii)* let $X, Y \in$ MOD.

As usual with Defeasible Logic, we have proof conditions for the monotonic part of the theory (proofs for the tagged literals $\pm\Delta^Y p$) and for the non-monotonic part (proofs for the tagged literals $\pm\partial^Y p$). To check licenses' compatibility and compose them means to apply the proof conditions of the logic to a multi-license where the set of licenses is $L = L_c$. Since the proof theory for the ontology component ($\pm\Delta^c p$ and $\pm\partial^c p$) is the one for standard Defeasible Logic we will omit it and refer the reader to [2]. For $\# \in \{\Delta, \partial\}$ and $Y \in \{\mathrm{O}, \mathrm{P}, \mathrm{F}\}$, notice that conditions governing conclusions for the composite license l_c and for any each license l_i interplay recursively: indeed, we may use a conclusion for l_c to fire a rule in l_i.

3.3 Provability in Each License

Definite Provability. The definitions below for Δ describe just forward (monotonic) chaining of strict rules.

Obligation Definite Provability

$+\Delta^{\mathrm{O}^{l_i}}$: If $P(n+1) = +\Delta^{\mathrm{O}^{l_i}} q$ then,
(1) $\mathrm{O}q \in F$ or
(2) $\exists r \in R_s^{\mathrm{O}^{l_i}}[q]$:
$\forall a, Xb, \neg Yd \in A(r)$:
$+\Delta^c a, +\Delta^{X^{l_c}} b, -\Delta^{Y^{l_c}} d \in P(1..n)$

$-\Delta^{\mathrm{O}^{l_i}}$: If $P(n+1) = -\Delta^{\mathrm{O}^{l_i}} q$ then
(1) $\mathrm{O}q \notin F$ and
(2) $\forall r \in R_s^{\mathrm{O}^{l_i}}[q]$:
$\exists a \in A(r)$: $-\Delta^c a \in P(1..n)$ or
$\exists Xb \in A(r)$: $-\Delta^{X^{l_c}} b \in P(1..n)$ or
$\exists \neg Yd \in A(r)$: $+\Delta^{Y^{l_c}} d \in P(1..n)$

Definite Provability for Prohibitions and Permissions. Definite proof conditions for prohibitions can be simply obtained from the ones for O.

$$\pm\Delta^{\mathrm{F}^{l_c}}: \text{ If } P(n+1) = \pm\Delta^{\mathrm{F}^{l_c}} q, \text{ then } \pm\Delta^{\mathrm{O}^{l_c}} \sim q \in P(1..n).$$

The concept of permission is much more elusive (for a discussion, see, e.g., [17]). Here, we minimize complexities by adopting perhaps the simplest option among those discussed in [11]. Such an option models permissive norms with defeaters for obligations: a defeater like $a_1, \ldots, a_n \rightsquigarrow^{l}_{\mathrm{O}} q$ states that some q is permitted (Pq) in the license l, since it is meant to block deontic defeasible rules for $\sim q$, i.e., rules supporting O$\sim q$[24]. This reading suggests that permissions are only defeasible, hence we postpone the proof theory for permission to the section dealing with the non-monotonic part of the theory[25].

Defeasible Provability. As usual in standard Defeasible Logic, to show that a literal q is defeasibly provable we have two choices: (1) we show that q is already definitely provable; or (2) we need to argue using the defeasible part of a multi-license theory D. For this second case, some (sub)conditions must be satisfied. First, we need to consider possible reasoning chains in support of $\sim q$ with the modes l_c and X^{l_c}, and show that $\sim q$ is not definitely provable with that mode (2.1 below). Second, we require that there must be a strict or defeasible rule with mode at hand for q which can apply (2.2 below). Third, we must consider the set of all rules which are not known to be inapplicable and which permit to get $\sim q$ with the mode under consideration (2.3 below). Essentially, each rule s of this kind attacks the conclusion q. To prove q, s must be counterattacked by a rule t for q with the following properties: i) t must be applicable, and ii) t must prevail over s. Thus each attack on the conclusion q must be counterattacked by a stronger rule. In other words, r and the rules t form a team (for q) that defeats the rules s.

Obligation Defeasible Provability

$+\partial^{\mathrm{O}^{l_i}}$: If $P(n+1) = +\partial^{\mathrm{O}^{l_i}} q$ then
(1) $+\Delta^{\mathrm{O}^{l_i}} q \in P(1..n)$ or
(2) (2.1) $-\Delta^{\mathrm{O}^{l_i}} \sim q \in P(1..n)$ and
(2.2) $\exists r \in R^{\mathrm{O}^{l_i}}_{\mathrm{sd}}[q]: \forall a, Xb, \neg Yd \in A(r)$: $+\partial^{c} a, +\partial^{X^{l_i}} b, -\partial^{Y^{l_i}} d \in P(1..n)$ and
(2.3) $\forall l_j \in \mathrm{Lic}, \forall s \in R^{\mathrm{O}^{l_j}}[\sim q]$, either
(2.3.1) $\exists a \in A(s)$ or $Xb \in A(s)$ or $\neg Y \in A(s)$:
$-\partial^{c} a \in P(1..n)$, or $-\partial^{X^{l_c}} b \in P(1..n)$, or $+\partial^{Y^{l_c}} d \in P(1..n)$; or
(2.3.2) $\forall l_k \in \mathrm{Lic}, \exists t \in R^{\mathrm{O}^{l_k}}[q]$: $\forall a, Xb, \neg Yd \in A(t)$,
$+\partial^{c} a, +\partial^{l_c} b, -\partial^{l_c} d \in P(1..n)$, and $t \succ s$.

[24] Hence, we do not make explicit in the language the distinction between the cases where we have explicit permissive clauses for P (strong permissions of q [25]) from those where some q is permitted (Pq) because it can be obtained from the fact that $\neg q$ is not provable as mandatory (weak permission). For an extensive treatment of defeasible permissions, see also [10].

[25] For space reasons, we will omit the proof conditions for $-\partial^{\mathrm{O}^{l_i}}$, and $-\partial^{\mathrm{P}^{l_i}}$, which can all be obtained applying the so-called Principle of Strong Negation [11], as illustrated for $-\Delta^{\mathrm{O}^{l_i}}$.

$+\partial^{\mathrm{P}^{l_i}}$: If $P(n+1) = +\partial^{\mathrm{P}^{l_i}} q$ then
(1) (1.1) $-\Delta^{\mathrm{O}^{l_i}} {\sim}q \in P(1..n)$ and
(1.2) $\exists r \in R^{\mathrm{O}^{l_i}}_{\mathrm{dft}}[q] : \forall a, Xb, \neg Yd \in A(r)$: $+\partial^c a, +\partial^{X^{l_i}} b, -\partial^{Y^{l_i}} d \in P(1..n)$ and
(1.3) $\forall l_j \in \mathrm{Lic}, \forall s \in R^{\mathrm{O}^{l_j}}[{\sim}q]$, either
(1.3.1) $\exists a \in A(s)$ or $Xb \in A(s)$ or $\neg Y \in A(s)$:
$-\partial^c a \in P(1..n)$, or $-\partial^{X^{l_c}} b \in P(1..n)$, or $+\partial^{Y^{l_c}} d \in P(1..n)$; or
(1.3.2) $\forall l_k \in \mathrm{Lic}, \exists t \in R^{\mathrm{O}^{l_k}}_{\mathrm{dft}}[q]$: $\forall a, Xb, \neg Yd \in A(t)$,
$+\partial^c a, +\partial^{l_c} b, -\partial^{l_c} d \in P(1..n)$, and $t \succ s$.

Let us consider two examples that illustrate some aspects of the proof theory, and how the heuristics are used before to formally introduce them.

Example 1. Assume to work with two licenses l_1 and l_2 and their composition, and let us reason only about obligations and permissions:

$$
\begin{aligned}
F &= \{a, d\} \\
R^{\mathrm{O}^{l_1}} &= \{r_1 : a \Rightarrow^{l_1}_{\mathrm{O}} p, \qquad r_2 : d \Rightarrow^{l_1}_{\mathrm{O}} {\sim}e\} \\
R^{\mathrm{O}^{l_2}} &= \{r_3 : \Rightarrow^{l_2}_{\mathrm{O}} {\sim}p, \qquad r_4 : a, d \rightsquigarrow^{l_2}_{\mathrm{O}} p, \qquad r_5 : \mathrm{P}p \Rightarrow^{l_2}_{\mathrm{O}} {\sim}e\} \\
{\succ} &= \{r_4 \succ r_3\}
\end{aligned}
$$

Let us consider AND-composition heuristics only. Rule r_1 leads in l_1 to $+\partial^{\mathrm{O}^{l_1}} p$ (i.e., Op in l_1). License l_2 supports $+\partial^{\mathrm{P}^{l_2}} p$ because the defeater r_4 is applicable and is stronger than r_3: hence, AND-composition states that $+\partial^{\mathrm{P}^{l_c}} p$ is the case (i.e., that p is permitted in the composite license). This last conclusion triggers r_5 thus obtaining in l_2 the conclusion $+\partial^{\mathrm{O}^{l_2}} {\sim}e$, the same deontic conclusion that is also obtained in l_1 by successfully applying r_2: hence, $+\partial^{\mathrm{O}^{l_c}} {\sim}e$ (i.e., e is prohibited in l_c).

Example 2. Consider two software libraries associated to licenses l_1 and l_2, respectively. License l_1 permits *Commercial* and obliges for *Attribution*, while license l_2 prohibits *Commercial*, permits *Derivative*, and obliges for *ShareAlike*.

$$
\begin{aligned}
L &= \{l_1, l_2\} \\
R^{\mathrm{O}^{l_1}} &= \{r_1 : \Rightarrow^{l_1}_{\mathrm{O}} \mathit{Attribution}, \qquad r_2 : \rightsquigarrow^{l_1}_{\mathrm{O}} \mathit{Commercial}\} \\
R^{\mathrm{O}^{l_2}} &= \{r_3 : \Rightarrow^{l_2}_{\mathrm{O}} {\sim}\mathit{Commercial}, \quad r_4 : \Rightarrow^{l_2}_{\mathrm{O}} \mathit{ShareAlike}, \qquad r_5 : \rightsquigarrow^{l_2}_{\mathrm{O}} \mathit{Derivative}\}
\end{aligned}
$$

We have to decide which heuristics better suits our needs with respect to the single licenses to compose. If we do not include the obligations present in each single license (*Attribution, ShareAlike*), we are not compliant with their normative semantics thus we violate them. To avoid that, the OR heuristics is used to compose obligations. Concerning permissions (*Derivative, Commercial*), we must check that every single license includes the specific permission, thus we adopt the AND heuristics. Otherwise, if there is a prohibition (${\sim}$*Commercial*), and we include the permission in l_c, we violate it. Hence, $+\partial^{\mathrm{O}^{l_c}} \mathit{Attribution}$, $+\partial^{\mathrm{O}^{l_c}} \mathit{ShareAlike}$, and $+\partial^{\mathrm{P}^{l_c}} \mathit{Derivative}$.

The proof conditions for composite licenses we define in the next section assume appropriate definitions for establishing whenever a deontic effect is entailed in a given license, as we presented.

3.4 Provability for Composite Licenses

According to the OR-composition and AND-composition heuristics, we may compose the deontic effects when each of them is entailed either by at least one license or by all licenses. This idea is directly captured as follows:

OR-composition For $\# \in \{\Delta, \partial\}$ and $X \in \{\mathrm{O}, \mathrm{P}, \mathrm{F}\}$:
$+\#^{X^{l_c}}$: If $P(n+1) = +\#^{X^{l_c}} p$, then $\exists l_i \in \mathrm{Lic} : +\#^{X^{l_i}} p \in P(1..n)$.
$-\#^{X^{l_c}}$: If $P(n+1) = -\#^{X^{l_c}} p$, then $\forall l_i \in \mathrm{Lic} : -\#^{X^{l_i}} p \in P(1..n)$.

AND-composition For $\# \in \{\Delta, \partial\}$ and $X \in \{\mathrm{O}, \mathrm{F}\}$:
$+\#^{X^{l_c}}$: If $P(n+1) = +\#^{X^{l_c}} p$, then $\forall l_i \in \mathrm{Lic} : +\#^{X^{l_i}} p \in P(1..n)$.
$-\#^{X^{l_c}}$: If $P(n+1) = -\#^{X^{l_c}} p$, then $\exists l_i \in \mathrm{Lic} : -\#^{X^{l_i}} p \in P(1..n)$.
$+\partial^{\mathrm{P}^{l_c}}$: If $P(n+1) = +\partial^{\mathrm{P}^{l_c}} p$, then $\exists l_i \in \mathrm{Lic} : +\partial^{\mathrm{P}^{l_i}} p \in P(1..n)$ and
$\forall l_k \in \mathrm{Lic}, l_i \neq l_k$ either $+\partial^{\mathrm{P}^{l_k}} p \in P(1..n)$ or $-\partial^{\mathrm{O}^{l_k}} \sim p \in P(1..n)$.
$-\partial^{\mathrm{P}^{l_c}}$: If $P(n+1) = -\partial^{\mathrm{P}^{l_c}} p$, then $\forall l_i \in \mathrm{Lic}$ either $-\partial^{\mathrm{P}^{l_i}} p$ or $+\partial^{\mathrm{P}^{l_i}} \sim p$.

The conditions for obligations and prohibitions directly implement what we have informally said in regard to the two heuristics. A brief comment about permissions in AND-composition: we may establish here that some p is permitted in l_c when it is explicitly permitted (via defeaters) in all licenses, or when there is at least one license explicitly permitting p and, in all the other licenses where no explicit permission for p succeeds, p is at least not prohibited (so p is weakly permitted [25,10]).

3.5 Properties and Admissibility

The logic presented here is a variant of the one developed in [8,9]. On account of this fact, two results can be imported here: its soundness and computational complexity.

Theorem 1. *Let D be a multi-license theory where the transitive closure of $\succ$ is acyclic. For every $\# \in \{\Delta, \partial\}$, $X \in \{l, Y^l | l \in \mathrm{Lic}, Y \in \{\mathrm{O}, \mathrm{F}\}\}$, and $Z \in \{l, W^l | l \in \mathrm{Lic}, W \in \mathrm{MOD}\}$:*

- *It is not possible that both $D \vdash +\#^Z p$ and $D \vdash -\#^Z p$;*
- *For all $l \in L \cup \{l_c\}$, it is not possible that both $D \vdash +\partial^{\mathrm{O}^l} p$ and $D \vdash +\partial^{\mathrm{P}^l} \sim p$;*
- *If $D \vdash +\partial^X p$ and $D \vdash +\partial^X \sim p$, then $D \vdash +\Delta^X p$ and $D \vdash +\Delta^X \sim p$.*

Given a multi-license theory D, the *universe* of D (U^D) is the set of all the atoms occurring in D. The *extension* (or conclusions) E^D of D is a structure $(\Delta_D^+, \Delta_D^-, \partial_D^+, \partial_D^-)$, where for $X^l \in \mathrm{MOD}$ and $l \in L$:

$$\Delta_D^{\pm} = \{Xq : D \vdash \pm\Delta^{X^l} q\} \cup \{q : D \vdash \pm\Delta^c q\} \qquad \partial_D^{\pm} = \{Xq : D \vdash \pm\partial^{X^l} q\} \cup \{q : D \vdash \pm\partial^c q\}.$$

Theorem 2. *Let $D = (F, L, R^c, \{R^{\mathrm{O}^l}\}_{l \in \mathrm{Lic}}, \succ)$ be a multi-license theory. The extension of D can be computed in time linear to the size of the theory, i.e., $O(|R^c \cup \{R^{\mathrm{O}^l}\}_{l \in \mathrm{Lic}}| * |U^D| * |L|)$.*

Finally, let us establish when a license composition l_c is meaningful or admissible. This can be checked taking into account the following guidelines:

- When only defeasible rules and defeaters are considered, a composition is admissible *iff* it leads to a non-empty set of deontic conclusions. Defeasible Logic is skeptical logic, so in case there is no way to solve deontic conflicts (according to any given heuristics), it means that the composite license does not produce any effect.
- In the case of conflicting strict rules there is no way to block contradictory conclusions. Hence, checking if a composition is admissible also requires to exclude that Δ_D^+ contains contradictory conclusions.
- Facts are supposed to describe a given situation where licenses are applied, thus they can vary from context to context. Hence, we may have two levels for detecting unsolvable conflicts in the licenses' composition: when we consider specific sets of facts, or when we examine licenses in general.

The following definition formally considers all these aspects:

Definition 2. *Let* $D = (F, L, \{R^l\}_{l\in \mathrm{Lic}}, \{R^{\mathrm{O}^l}\}_{l\in \mathrm{Lic}}, \{R^{\mathrm{P}^l}\}_{l\in \mathrm{Lic}}, \succ)$ *be a multi-license theory and* $\mathscr{A}_D$ *is the set of all literals and modal literals occurring in the antecedent of all rules of D. The license* $l_c = l_1 \odot \cdots \odot l_n$ *is* F-admissible *iff*

- $L = \{l_1, \ldots, l_n\}$,
- $\exists X^{l_c} q \in \partial_D^+$, *and*
- *for any literal p, if* $X \in \{l, Y^l | l \in \mathrm{Lic}, Y \in \{\mathrm{O}, \mathrm{F}\}\}$, *then we do not have that* $D \vdash +\Delta^X p$ *and* $D \vdash +\Delta^X \sim p$.

The composite license $l_c = l_1 \odot \cdots \odot l_n$ *is* admissible *iff it is* F*-admissible for all* $F \subseteq \mathscr{A}_D$.

4 Mapping into SPINdle and Results

In this section we illustrate how to implement the logic and the heuristics developed in Section 3 in SPINdle[26]. SPINdle [15] is a modular and efficient reasoning engine, written in Java, for defeasible logic and modal defeasible logic implementing and extending the algorithms of [8,9]. It has been experimentally tested against the benchmark of [16] showing that it is able to handle very large theories, i.e., theories with hundredth of thousand rules, indeed the largest theory it has been tested with has 1 million rules.

While SPINdle supports multi-modal defeasible logics, currently it does not support natively the AND and OR heuristics presented in this paper. Therefore, we first have to provide polynomial time transformations to implement the two heuristics.

Definition 3. *Let* # *be one of the proof tags. Two multi-license theories* D_1 *and* D_2 *are equivalent (written* $D_1 \equiv D_2$*) iff* $\forall p, D_1 \vdash \#p$ *iff* $D_2 \vdash \#p$, *i.e., they have the same consequences. Similarly* $D_1 \equiv_\Sigma D_2$ *means that* D_1 *and* D_2 *have the same consequences in the language* Σ.

Definition 4. *A* transformation *is a mapping from multi-license theories to multi-license theories. A transformation T is* correct *iff for all theories* D_i, $D \equiv_\Sigma T(D)$ *where* Σ *is the language of D.*

[26] `http://spin.nicta.org.au/spindle/index.html`

The OR-heuristic is implemented by the following transformation[27]:

$$tor(r) = \begin{cases} r\colon A(r) \hookrightarrow p & r \in R^c \\ r\colon A(r) \rightarrow_{\mathrm{O}^c} p & r \in R_{\mathrm{s}}^{\mathrm{O}^{l_i}},\ l_i \in \mathrm{Lic} \\ r\colon A(r) \Rightarrow_{\mathrm{O}^c} p & r \in R_{\mathrm{d}}^{\mathrm{O}^{l_i}},\ l_i \in \mathrm{Lic} \\ r\colon A(r) \Rightarrow_{-\mathrm{O}^c} {\sim}p & r \in R_{\mathrm{dft}}^{\mathrm{O}^{l_i}},\ l_i \in \mathrm{Lic} \end{cases}$$

It is immediate to see that *tor* is a one-to-one transformation. For obligation operators *tor* flattens all of them into O^c. In fact, for the OR-heuristics we need to prove p with $+\partial^{\mathrm{O}^{l_i}}$ for a single license, and thus the set of rules to be considered for clause (2.2) is the set of all strict and defeasible obligation rules for p. For permission we use a particular feature of SPINdle, namely 'negative' modalities. A negative modality (e.g. $-\mathrm{O}^c$) behaves on one side as any other modality, but it is in a symmetric conflict with the corresponding positive one (i.e., O^c). Thus it can be used to disprove a conclusion for the positive counterpart without proving it. Hence, it behaves essentially like a defeater. Theorem 1 shows that the transformation of *tor* into the logic of SPINdle is correct.

Theorem 1. *Let $D = (F, L, R, \succ)$ be a multi-license theory. Let $T(D) = (F, L, \{tor(r) : r \in R\}, \succ)$. Then $D \equiv_{\Sigma} T(D)$.*

We show now by means of an example how to apply our logic to compose three Web of Data popular licenses using the SPINdle transformation.

Example 3. Assume the data returned by different datasets are associated to the Open Government License[28], the Open Database License[29], and the Attribution-NonCommercial-NoDerivs 2.0 Generic License[30]. The three licenses in a machine-readable format are visualized in Figure 2. The multi-license theory D is as follows:

$$\begin{aligned}
F &= \{Open\} \\
L &= \{l_{OGL}, l_{ODbL}, l_{BY-NC-ND}\} \\
R^{\mathrm{O}^{l_{OGL}}} &= \{r_1 :\Rightarrow_{\mathrm{O}}^{l_{OGL}} Attribution, && r_2 : Open \rightsquigarrow_{\mathrm{O}}^{l_{OGL}} Publishing, \\
&\quad r_3 : Open \rightsquigarrow_{\mathrm{O}}^{l_{OGL}} Distribution, && r_4 : Open \rightsquigarrow_{\mathrm{O}}^{l_{OGL}} Derivative, \\
&\quad r_5 : Open \rightsquigarrow_{\mathrm{O}}^{l_{OGL}} Commercial\} \\
R^{\mathrm{O}^{l_{ODbL}}} &= \{r_6 :\Rightarrow_{\mathrm{O}}^{l_{ODbL}} ShareAlike, && r_7 :\Rightarrow_{\mathrm{O}}^{l_{ODbL}} Attribution, \\
&\quad r_8 :\rightsquigarrow_{\mathrm{O}}^{l_{ODbL}} Sharing, && r_9 :\rightsquigarrow_{\mathrm{O}}^{l_{ODbL}} Derivative\} \\
R^{\mathrm{O}^{l_{BY-NC-ND}}} &= \{r_{10} :\Rightarrow_{\mathrm{O}}^{l_{BY-NC-ND}} Attribution, && r_{11} :\Rightarrow_{\mathrm{O}}^{l_{BY-NC-ND}} {\sim}Commercial, \\
&\quad r_{12} :\Rightarrow_{\mathrm{O}}^{l_{BY-NC-ND}} {\sim}Derivative, && r_{13} :\rightsquigarrow_{\mathrm{O}}^{l_{BY-NC-ND}} Sharing\} \\
\succ &= \{l_{ODbL} \succ l_{BY-NC-ND}\}
\end{aligned}$$

[27] In the remainder, O^c and P^c abbreviate O^{l_c} and P^{l_c}.

[28] `http://www.nationalarchives.gov.uk/doc/open-government-licence/`

[29] `http://opendatacommons.org/licenses/odbl/`

[30] `http://creativecommons.org/licenses/by-nc-nd/3.0/`

We have now to build the composite license such that $l_c = l_{OGL} \odot l_{ODbL} \odot l_{BY-NC-ND}$. AND-composition is admissible since there is at least one deontic effect entailed by all licenses, i.e., from rules r_1, r_7 and r_{10}, which lead to the deontic conclusion $+\partial^{O^{l_c}} Attribution$. OR-composition is admissible too: notice that a conflict arises between rule r_5 and rule r_{11} and between rule r_{12} and rules r_4 and r_9. The deontic conclusions are: $+\partial^{O^{l_c}} Attribution$, $+\partial^{O^{l_c}} ShareAlike$, $+\partial^{P^{l_c}} Publishing$, $+\partial^{P^{l_c}} Distribution$, $+\partial^{P^{l_c}} Sharing$, $-\partial^{P^{l_c}} Derivative$, $-\partial^{P^{l_c}} Commercial$. The tor transformation is

```
>> Open                                  r8: =>[-Oc] -Share
r1: =>[Oc]Attribution                    r9: =>[-Oc] -Derivative
r2: Open =>[-Oc] -Publishing
r3: Open =>[-Oc] -Distribution           r10: =>[Oc] Attribution
r4: Open =>[-Oc] -Derivative             r11: =>[Oc] -CommercialExpl
r5: Open =>[-Oc] -CommercialExpl         r12: =>[Oc] -Derivative
                                         r13: =>[-Oc] -Share
r6: =>[Oc] ShareAlike
r7: =>[Oc] Attribution                   r9 > r12
```

When the above theory is loaded in SPINdle it takes 14 milliseconds to produce the following conclusions `+d [Oc]Attribution`, `+d [-Oc]-Distribution`, `+d [-Oc]-Publishing`, `+d [-Oc]-Share`, `+d [Oc]ShareAlike`, where `+d [Oc]` corresponds to $+\partial^{O^c}$, and `+d [-Oc]-` means $+\partial^{P^c}$. Figure 2.d shows the machine-readable l_c.

```
@prefix l4lod: http://ns.inria.fr/l4lod/.
@prefix : http://example/licenses.

:licOGL a l4lod:License;
       l4lod:licensingTerms <http://www.nationalarchives.gov.uk/
                                 doc/open-government-licence/>;
       l4lod:permits l4lod:Publishing;
       l4lod:permits l4lod:Distribution;
       l4lod:permits l4lod:Derivative;
       l4lod:permits l4lod:CommercialExpl;
       l4lod:obliges l4lod:Attribution.
```
(a)

```
@prefix l4lod: http://ns.inria.fr/l4lod/.
@prefix : http://example/licenses.

:licODbL a l4lod:License;
         l4lod:licensingTerms <http://opendatacommons.org/
                                        licenses/odbl/>;
         l4lod:permits l4lod:Sharing;
         l4lod:permits l4lod:Derivative;
         l4lod:obliges l4lod:Attribution;
         l4lod:obliges l4lod:ShareAlike.
```
(b)

```
@prefix cc: http://creativecommons.org/ns.
@prefix l4lod: http://ns.inria.fr/l4lod/.
@prefix : http://example/licenses.

:licBY-CC-NC-ND a cc:License;
          cc:legalcode <http://creativecommons.org/licenses
                                              /by-nc-nd/>;
          cc:permits cc:Sharing;
          cc:requires cc:Attribution;
          cc:prohibits cc:CommercialUse;
          l4lod:prohibits l4lod:NoDerivative.
```
(c)

```
@prefix l4lod: http://ns.inria.fr/
l4lod/.
@prefix : http://example/licenses.

:licComposite a l4lod:License;
    l4lod:obliges l4lod:Attribution;
    l4lod:obliges l4lod:ShareAlike;
    l4lod:permits l4lod:Publishing;
    l4lod:permits l4lod:Distribution;
    l4lod:permits l4lod:Sharing.}
```
(d)

Fig. 2. Licenses to be composed (a-b-c) and the resulting composite license (d)

Notice that the overhead introduced by the licenses composition framework is constituted by the query execution time to retrieve the licenses associated to the triples returned as query result (if the set of licenses is known, it can be pre-computed), plus the SPINdle overhead in computing the composite license. For these reasons, we can say that the actual overhead is represented by SPINdle but, as shown above, it does not have a serious impact on query execution time (few milliseconds).

For the AND-heuristic, for a multi-license theory $D = (F, L = \{l_1, \ldots, l_n\}, R, \succ)$, the transformation is based on the following sets of rules:

$$tand(r) = \{r_{ij} : A(r) \rightsquigarrow_{\mathrm{O}^j} C(r) | r \in R^{\mathrm{O}^{l_i}}\} \cup \{r | r \in R^{\mathrm{O}^{l_i}}_{\mathrm{sd}}\} \cup \{r : A(r) \Rightarrow_{-\mathrm{O}^i} C(r) | r \in R^{\mathrm{O}^{l_i}}_{\mathrm{dft}}\}$$

$$R^* = \{o_q : \mathrm{O}^1 q, \ldots, \mathrm{O}^n q \Rightarrow_{\mathrm{O}^c} q, \quad p_q : \mathrm{P}^* q, \mathrm{P}^1 q, \ldots, \mathrm{P}^n q \Rightarrow_{\mathrm{P}^c} q$$
$$p^{i*}_q : -\mathrm{O}^i \sim q \Rightarrow_{\mathrm{P}^*} q, \quad p^i_q : -\mathrm{O}^i \sim q \Rightarrow_{\mathrm{P}^i} q, \quad f^i_q : \neg \mathrm{O}^i q \Rightarrow_{\mathrm{P}^i} q \mid$$
$$l_i \in L, \exists r \in R^{\mathrm{O}^{l_i}}, C(r) = q\}$$

Defeaters are modeled as in the *tor* transformation. The intuition behind the rules r_{ij} is that all rules can be use to attack any other rule, irrespectively of the license. Thus for every obligation rule in a license we create a defeater with the same content for each other license. For the AND-composition of obligations we need that a literal q is provable as obligation in every license; this is achieved by rule o_q. We have to do the same for permissions. However, permission requires that at least one license permits q and for all other licenses q is either permitted or not forbidden. To achieve this we create a 'special' modality P^* and rules linking this to provability of the negative modality $-\mathrm{O}^i$ (encoding permission in SPINdle). Finally, for each license we create its permission modality P^i and a literal q can be derived with such modality if it is permitted in license l_i ($-\mathrm{O}^i q$, rules p^i_q), or if q is not forbidden by that license (i.e., $\neg \mathrm{O}^i q$, rules f^i_q).

Theorem 2 shows that the transformation of *tand* into the logic of SPINdle is correct.

Theorem 2. *Let $D = (F, L, R, >)$ be a multi-license theory. $T(D) = (F, L, \{r | r \in R^c\} \cup \{tand(r) | r \in R^{\mathrm{O}^{l_i}}\}_{l_i \in \mathrm{Lic}} \cup R^*, \succ \cup \{(r_{ij}, s_{ij}), (r_{ij}, s), (r, s_{ij}) | r \succ s\}_{l_i, l_j \in \mathrm{Lic}})$. Then $D \equiv_\Sigma T(D)$.*

5 Related Work

The logic we presented is an extension of the logic of [8,9]. The debt on previous work is the general idea of the formalism, and the proof theory for obligations. What is new is the way for composing licenses (Section 3.4), and for computing permissions and relative results (Section 3.5). Also, we proved that there is a transformation mapping the new logic into SPINdle.

Pucella and Weissman [20] propose a logic to check whether the user's actions follow the licenses' specifications. They do not deal with composition and do not provide a deontic account of licenses' conclusions. Furthermore, their logic is not able to handle conflicting licenses.

Nadah et al. [19] propose to assist licensors' work by providing them a generic way to instantiate licenses, independent from specific formats. We go towards the definition of a composite license while they go towards the definition of a specific ontology (about 100 concepts) used for the translation in the different formats.

Gangadharan et al. [6] address the issue of service license composition and compatibility analysis basing on ODRL-S, an extension of ODRL to implement the clauses of service licensing. They specify a matchmaking algorithm which verifies whether two service licenses are compatible. In case of a positive answer, the services can be composed and the framework determines the license of the composite service. Truong et al. [22] address the issue of analyzing data contracts, based on ODRL-S. Contract analysis leads to the definition of a contract composition where first the comparable contractual terms from the different data contracts are retrieved, and second an evaluation of

the new contractual terms for the data mash-up is addressed. Villata and Gandon [23] follow a similar approach to evaluate the compatibility of CC licenses and compose them. There are several differences w.r.t. these approaches: *(i)* the application scenario is different (service composition vs. Web of Data); *(ii)* we allow for a normative reasoning which goes beyond basic compatibility rules by exploiting normative compliance. However, common points are the idea of merging the clauses of the different licenses/-contracts, and the use of RDF for licenses/contracts representation.

Krotzsch and Speiser [14] present a semantic framework for evaluating ShareAlike recursive statements. In particular, they develop a general policy modelling language, then instantiated with OWL DL and Datalog, for supporting self-referential policies as expressed by CC. In this paper, we address another kind of problem that is the composition of the deontic components of single licenses into a composite license.

Gordon [7] presents a legal prototype for analyzing open source licenses compatibility using the Carneades argumentation system. Licenses compatibility is addressed at a different granularity w.r.t. our purpose, and licenses composition is not considered.

The attachment of additional information like rights or licenses to RDF triplets is linked to an active research field. Carroll et al. [5] introduced Named Graphs in RDF to allow publishers to communicate assertional intent and to sign their assertions. Moreover, the W3C Provenance WG defines the kind of information to be used to form assessments about data quality, reliability or trustworthiness [12].

6 Conclusions

In this paper, we propose an automated framework for licenses composition based on deontic logic. The rationale behind this framework is to build a composite license starting from the single licensing terms associated to heterogeneous data. We adopt deontic logic to ensure the compliance of the composite license with respect to the single licenses composing it. We evaluate the feasibility of the automatic generation of the composite license on the SPINdle defeasible reasoner.

There are several lines to pursue as future research. First, we will develop a standalone licensing module generating the machine-readable composite license every time a query returns multi-licensed data. Second, we still have to consider the case of data obtained by inference from one or several licensed datasets. In particular, a special case we have to address is the one of queries going beyond basic `SELECT` queries, where aggregations are present, e.g., return the *average*, *sum*, etc. of the data possibly over distributed datasets. Third, the logic should take into account the temporal aspect of licenses. In particular, two concepts to be considered are *validity time* (point in time where a deontic component is true) and *reference time* (point in time the obligation, prohibition or permission applies to) of an obligation, prohibition or permission. Finally, even if our framework allows to reason about certain characteristics of licenses, e.g., whether attribution is required or commercial usages are permitted, it is still an open problem the fact that there is no uniform, cross-national definition of essential legal terms. We will investigate suitable solutions with further legal experts.

References

1. Abelson, H., Adida, B., Linksvayer, M., Yergler, N.: ccREL: The creative commons rights expression language. Tech. rep. (2008)

2. Antoniou, G., Billington, D., Governatori, G., Maher, M.J.: Representation results for defeasible logic. ACM Transactions on Computational Logic 2(2), 255–287 (2001)
3. Bizer, C., Heath, T., Berners-Lee, T.: Linked data - the story so far. Int. J. Semantic Web Inf. Syst. 5(3), 1–22 (2009)
4. Boella, G., Governatori, G., Rotolo, A., van der Torre, L.: A logical understanding of legal interpretation. In: Proceedings of KR. AAAI Press (2010)
5. Carroll, J., Bizer, C., Hayes, P., Stickler, P.: Named graphs. J. Web Sem. 3(4), 247–267 (2005)
6. Gangadharan, G.R., Weiss, M., D'Andrea, V., Iannella, R.: Service license composition and compatibility analysis. In: Krämer, B.J., Lin, K.-J., Narasimhan, P. (eds.) ICSOC 2007. LNCS, vol. 4749, pp. 257–269. Springer, Heidelberg (2007)
7. Gordon, T.F.: Analyzing open source license compatibility issues with Carneades. In: Proceedings of ICAIL, pp. 51–55. ACM (2011)
8. Governatori, G., Rotolo, A.: BIO logical agents: Norms, beliefs, intentions in defeasible logic. Autonomous Agents and Multi-Agent Systems 17(1), 36–69 (2008)
9. Governatori, G., Rotolo, A.: A computational framework for institutional agency. Artif. Intell. Law 16(1), 25–52 (2008)
10. Governatori, G., Olivieri, F., Rotolo, A., Scannapieco, S.: Three concepts of defeasible permission. In: Proceedings of JURIX, pp. 63–72. IOS Press (2011)
11. Governatori, G., Padmanabhan, V., Rotolo, A., Sattar, A.: A defeasible logic for modelling policy-based intentions and motivational attitudes. Logic J. of IGPL 17(3), 227–265 (2009)
12. Groth, P.T., Gil, Y., Cheney, J., Miles, S.: Requirements for provenance on the web. IJDC 7(1), 39–56 (2012)
13. Heath, T., Bizer, C.: Linked Data: Evolving the Web into a Global Data Space. Morgan & Claypool (2011)
14. Krötzsch, M., Speiser, S.: ShareAlike Your Data: Self-referential Usage Policies for the Semantic Web. In: Aroyo, L., Welty, C., Alani, H., Taylor, J., Bernstein, A., Kagal, L., Noy, N., Blomqvist, E. (eds.) ISWC 2011, Part I. LNCS, vol. 7031, pp. 354–369. Springer, Heidelberg (2011)
15. Lam, H.P., Governatori, G.: The making of SPINdle. In: Governatori, G., Hall, J., Paschke, A. (eds.) RuleML 2009. LNCS, vol. 5858, pp. 315–322. Springer, Heidelberg (2009)
16. Maher, M.J., Rock, A., Antoniou, G., Billington, D., Miller, T.: Efficient defeasible reasoning systems. International Journal of Artificial Intelligence Tools 10, 483–501 (2001)
17. Makinson, D., van der Torre, L.: Permission from an input/output perspective. Journal of Philosophical Logic 32(4), 391–416 (2003)
18. Miller, P., Styles, R., Heath, T.: Open data commons, a license for open data. In: Proceedings of LDOW (2008)
19. Nadah, N., de Rosnay, M.D., Bachimont, B.: Licensing digital content with a generic ontology: escaping from the jungle of rights expression languages. In: Proceedings of ICAIL, pp. 65–69. ACM (2007)
20. Pucella, R., Weissman, V.: A logic for reasoning about digital rights. In: Proceedings of CSFW, pp. 282–294. IEEE (2002)
21. Rotolo, A., Villata, S., Gandon, F.: A deontic logic semantics for licenses composition in the web of data. In: Proceedings of ICAIL, pp. 111–120. ACM (2013)
22. Truong, H.L., Gangadharan, G.R., Comerio, M., Dustdar, S., Paoli, F.D.: On analyzing and developing data contracts in cloud-based data marketplaces. In: Proceedings of APSCC, pp. 174–181. IEEE (2011)
23. Villata, S., Gandon, F.: Licenses compatibility and composition in the web of data. In: Proceedings of COLD. CEUR Workshop Proceedings, vol. 905 (2012)
24. von Wright, G.: Deontic logic. Mind 60(237), 1–15 (1951)
25. von Wright, G.: Norm and action: A logical inquiry. Routledge and Kegan Paul (1963)

Federated Entity Search Using On-the-Fly Consolidation

Daniel M. Herzig[1,⋆], Peter Mika[2], Roi Blanco[2], and Thanh Tran[1]

[1] Karlsruhe Institute of Technology (KIT), Germany
[2] Yahoo! Research, Spain
{herzig,ducthanh.tran}@kit.edu, {roi,pmika}@yahoo-inc.com

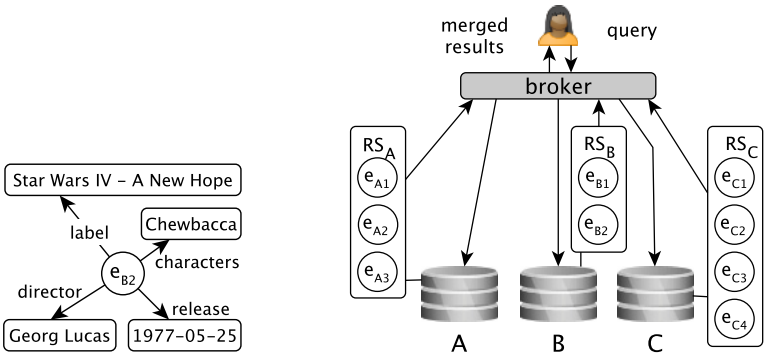
Abstract. Nowadays, search on the Web goes beyond the retrieval of textual Web sites and increasingly takes advantage of the growing amount of structured data. Of particular interest is entity search, where the units of retrieval are structured entities instead of textual documents. These entities reside in different sources, which may provide only limited information about their content and are therefore called "uncooperative". Further, these sources capture complementary but also redundant information about entities. In this environment of uncooperative data sources, we study the problem of *federated entity search*, where redundant information about entities is reduced on-the-fly through *entity consolidation* performed at query time. We propose a novel method for entity consolidation that is based on using language models and completely unsupervised, hence more suitable for this on-the-fly uncooperative setting than state-of-the-art methods that require training data. Further, we apply the same language model technique to deal with the federated search problem of ranking results returned from different sources. Particular novel are the mechanisms we propose to incorporate consolidation results into this ranking. We perform experiments using real Web queries and data sources. Our experiments show that our approach for federated entity search with on-the-fly consolidation improves upon the performance of a state-of-the-art preference aggregation baseline and also benefits from consolidation.

1 Introduction

Taking advantage of the growing amount of structured data on the Web has been recognized as a promising way to improve the effectiveness of search and has therefore gained the interest of researchers and industry [1]. This development is also driven by the demand from Web search users, whose most dominant search task is the search for entities. Recent studies showed that about 70% of Web search queries contain entities [2] and that the intent of about 40% of unique Web queries is to find a particular entity [3]. In contrast to named entities, which are text tokens identifying specific concepts, e.g. the name of a person, the increasing amount of structured data on the Web as well as the availability of knowledge bases allows to perceive entities not just as single tokens, but as structured objects with attributes and values, e.g. Figure 1a illustrates an entity representing the movie "Star WarsIV - A New Hope".

⋆ Work done while being visiting researcher at Yahoo! Labs, Barcelona.

H. Alani et al. (Eds.): ISWC 2013, Part I, LNCS 8218, pp. 167–183, 2013.



(a) A structured entity with four attributes and values

(b) The broker obtains only the ranked result lists RS_A, RS_B, RS_C from each data source and merges them into one single ranked list

Fig. 1. Federated Entity Search in an uncooperative setting

The entities reside in different sources across the Web or originate from different knowledge bases. These sources capture redundant but also complementary information about entities. Hence, consolidating co-referent entities referring to the same real-world object and providing *search functionalities over co-referent entities* is a crucial step towards exploiting structured data sources for retrieval. In particular, the need for large scale and fast coreference has been recognized and recently a solution in this direction has been proposed [4]. This solution and the comprehensive work of the database community in this realm [5–7] assume full access to the entire datasets to compute features such as weights of attributes, co-occurences or to learn parameters, which are then used to resolve all coreferences between two or more datasets in one run. However, access to the entire datasets is either not granted in many application scenarios such as search over multiple Web data sources (where data access is only provided via APIs for single requests), also called federated search over *uncooperative* sources [8, 9], or many data sources are highly dynamic, imposing a high burden on batch processing to keep up with frequent changes and to provide fresh information for time sensitive applications such as search over stock quotes, movies and timetables. Distributed document retrieval for uncooperative environments has been studied in the IR community [8, 9]. We investigate the task of federated entity search with structured entities consisting of a varying number of attributes and corresponding values. This task is different from document retrieval, where each document consists of exactly one text body.

Contributions. We address the setting of *uncooperative environments*, where neither prior information about the data sources nor training data is available and propose a language model (LM) based approach for *federated entity search in uncooperative environments* using query time entity consolidation.

We present three contributions: (1) We propose a LM based unsupervised approach for computing the similarity between entities and use it to perform query time entity consolidation. (2) We reuse these LM based representations of entities and we show how this entity representation in combination with composite relevance models [10, 11] can be used to obtain a combined ranking of results returned from different sources. The mechanisms we propose to incorporate consolidation results into this ranking are particularly novel. (3) In our experiments, we employ real-world Web queries and data sources and investigate the effects of federated search in combination with consolidation on retrieval performance. We show that our approach exceeds a state-of-the-art preference aggregation method for federated search [12] and show the advantages of consolidation for search in federated settings.

2 Overview

We follow the definition by Pound et al. [3] and define *entity search* as the task of answering arbitrary information needs related to particular aspects of entities, expressed in unconstrained natural language and resolved using a collection of structured data. We address a particular kind of entity search, namely the search over multiple data sources, called *federated search*, which entails the three main problems of *source representation*, *source selection*, and *result merging* [8, 9]. We focus on the latter for *federated entity search in uncooperative settings* as illustrated in Figure 1b, where only ranked result lists of entity descriptions are obtained from each source and no further information about the sources is available. In this scenario, we perform *consolidated entity search* where entities representing the same real-world object, called *co-referent entities* are identified, linked and incorporated into ranking to avoid redundant results. Further, Web data is heterogeneous in the sense that differences in schema and vocabulary are common. Coping with schema differences has been studied in the area of schema matching and for the given setting also in our previous work [11]. Here, it is considered as out of scope. We illustrate the addressed problem throughout the paper using the following example.

Example 1. Assume the keyword query *"star wars"* is issued to three data sources, which hold information about movies. Each source returns a ranked list of structured entities as illustrated in Figure 1b and the same lists are shown in more detail in Figure 2a. We observe that the lists contain co-referent entities, e.g. e_{B2} in RS_B and e_{C1} in RS_C both represent the same movie "A New Hope". If we put all entities in one result list, we obtain a list with redundant results. Later, we will refer to this case **w**ithout consolidation as CRM_w. In our example, we observe in Figure 2b that within the first six ranks only three distinct entities are shown, because $\{e_{B2}, e_{C3}, e_{A2}\}$ and $\{e_{C3}, e_{C2}\}$ are co-referent (indicated by identical line type). Our goal is to **c**onsolidate the results by grouping co-referent entities into sets as illustrated in Figure 2c, which we will later refer to as CRM_c. In particular, we will focus on ranking in this setting and investigate the effects on retrieval performance of federated search with and without consolidation.

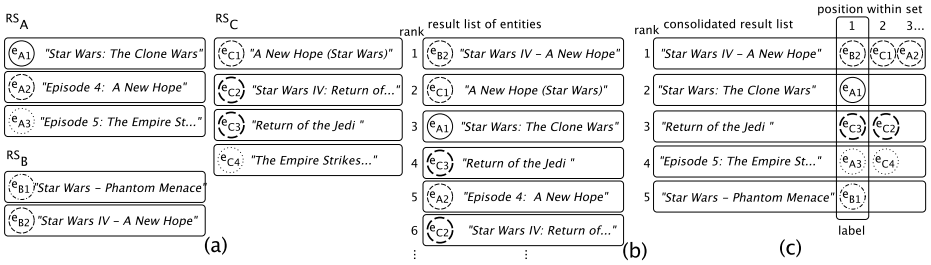


Fig. 2. (a) Result lists of each source. (b) Merged result list containing co-references without consolidation (CRM_w). (c) Consolidated result list with co-referent sets using the entity on position 1 as label (CRM_c).

We focus on Web data for which the RDF model has been proposed as a W3C standard for data representation and interchange. For the sake of generality, we omit RDF specific features, like blank nodes, and employ a general graph-structured data model.

A *data source* is a directed and labeled graph $G = (N, E)$. The set of nodes N is a disjoint union of *entities* N_E and literals N_L, i.e. $N = N_E \uplus N_L$. Edges E can be conceived as a disjoint union $E = E_E \uplus E_L$ of edges representing connections between entities, i.e. $a(e_i, e_j) \in E_E$, iff $e_i, e_j \in N_E$, and connections between entities and literals also called *attribute values*, i.e. $a(e, v) \in E_L$, iff $e \in N_E$ and $v \in N_L$. Given this graph G, we call the bag of attribute value edges $A(e) = \{a(e, v) \in E | v \in N_L\}$ the *description* of the entity $e \in N_E$, and each $a(e, v) \in A(e)$ is called an *attribute* of e. The set of distinct attribute labels of an entity e, i.e. $A'(e) = \{a | a(e, v) \in A(e)\}$, is called the *model* of e. Figure 1a illustrates entity e_{B2}, which has the model $A'(e_{B2}) = \{label, director, characters, release\}$. In our Web scenario, each data source is represented by a graph G_X, for example Figure 1b illustrates three data sources $X = \{A, B, C\}$. Although arbitrary edges can connect entities across data graphs, we are only interested in edges denoting that two entities are co-referent, i.e. $a_{\text{same}}(e_X, e_Y), e_X \in G_X, e_Y \in G_Y$, e.g. `owl:sameAs` [1].

3 On-the-fly Entity Consolidation

Entity consolidation is typically performed through the main steps of *representing entities* as attribute value pairs, and finding the appropriate *similarity metric* and *threshold* to determine whether two given entity representations refer to the same object or not, i.e. when the similarity computed using the metric exceeds the threshold. Since several attributes are typically used, state-of-the-art methods employ supervised machine learning techniques to learn the weights for attributes or also the metrics and thresholds [13]. In this section, we (1) represent attribute values as language models (LMs), (2) employ a specific notion

[1] `http://www.w3.org/TR/owl-ref/#sameAs-def`

of distance for LMs as the similarity metric, and (3) propose an unsupervised technique to estimate the weight associated with each attribute LM. In our approach, all the steps needed to derive the LM-based entity representation as well as the actual detection of coreferences are performed on-the-fly during the execution of a query.

3.1 Entity Representation

In entity search, composite LMs have been proposed to represent an entity as a collection of multinomial distributions, each capturing one particular entity attribute [11, 14] and are used to compute the similarity between an entity and a query. This modeling is suited when entities are not only associated with concise values but descriptions – which is the case for many Web data sources capturing entities via attributes such as *label* and *comment*. We use this representation also for the consolidation task.

The composite LM $\mathcal{P}$ for an entity e is constructed by considering all attributes in the model of e, $a \in A'(e)$. That is, $\mathcal{P}$ contains a LM $P_e(w|a)$ for every $a \in A'$. Every $P_e(w|a)$ captures the probability of observing a word w in the attribute values of a associated with e. Let $V_a(e)$ be the bag of value nodes of the attribute a associated with e, i.e. $V_a(e) = \{v|a(e,v) \in E, v \in N_L\}$, and w a word in the vocabulary, then $P_e(w|a)$ is estimated using maximum-likelihood as follows:

$$P_e(w|a) = \frac{\sum_{v \in V_a(e)} n(w,v)}{\sum_{v \in V_a(e)} |v|} \tag{1}$$

where $n(w,v)$ denotes the count of w in the value v, and $|v|$ is the total number of words in v.

3.2 Similarity Metric

Given two entities in the result lists, $e_X \in RS_X$ and $e_Y \in RS_Y$, we determine whether they are co-referent or not using a similarity metric. Standard metrics used by consolidation methods include *edit distance* and *Jaccard similarity*, which can be applied to two given attribute values. The former captures the number of edit operations needed to transform one value to the other while the latter is based on the word overlaps between the two values. Since we apply LMs to captures values, we measure the overlap of the LMs with the *Jensen-Shannon divergence* (JSD), which is based on the *Kullback-Leibler divergence* (KLD). The JSD however has the advantages of being symmetric, bounded ($0 \leq JSD \leq 1$), smoothed and its square root is a metric. Given the probability distributions P_X, P_Y and $R = \frac{1}{2}P_X + \frac{1}{2}P_Y$, the JSD is defined as:

$$JSD(P_X||P_Y) = \frac{1}{2}KLD(P_X||R) + \frac{1}{2}KLD(P_Y||R) \tag{2}$$

where $KLD(P||R) = \sum_w P(w) \log_2 \frac{P(w)}{R(w)}$. For computing the distance d between two entities e_X and e_Y, $d(e_X, e_Y)$, we use the square root of the JSDs calculated

over the LMs constructed for all attributes a that both entities have in common and weight each overlap measured by the JSD by $\omega(a)$:

$$d(e_X, e_Y) = \frac{1}{\sum_{a \in A'(e_X) \cap A'(e_Y)} \omega(a)} \sum \omega(a)\ JSD(P_{e_X}(w|a)||P_{e_Y}(w|a))^{\frac{1}{2}} \tag{3}$$

3.3 Estimating Weights

The weight $\omega(a)$ expresses how discriminative and identifying an attribute a is. We determine $\omega(a)$ w.r.t. the result lists RS_X and RS_Y. First, we construct a LM $P_X(w|a)$ analog to Equation 1. However, $P_X(w|a)$ captures the values of all the entities in RS_X instead of a single entity, i.e. $V_a(RS_X) = \{v|a(e_X, v), e_X \in RS_X\}$ instead of $V_a(e)$. Then, we compute the entropy $H(P) = -\sum_w P(w) \log_2 P(w)$ and set $\omega(a)$ to:

$$\omega(a) = \frac{1}{2} H(P_X) H(P_Y) \tag{4}$$

The rationale behind this formulation is that the entropy is high, if the bag $V_a(RS_X)$ contains many diverse values, and it is low, if $V_a(RS_X)$ contains similar and hence less discriminative values. Attributes with more diverse values are associated with higher weights because they provide more discriminate information to distinguish entities.

3.4 Entity Similarity

Given the above distance function and two result lists $RS_X = (e_{X1}, e_{X2},..., e_{Xi},...)$ and $RS_Y = (e_{Y1}, e_{Y2},..., e_{Yj},...)$, we consider two entities $e_X \in RS_X$ and $e_Y \in RS_Y$ as co-referent, if their distance is below a threshold t and if they are mutually the closest to each other, see Equation 5. The latter condition assures that only co-references are established, if a candidate entity is favored over all alternatives. Note, that we also compare the sources to themselves, i.e. $X = Y$, to find co-references within a source.

$$\begin{aligned} &d(e_X, e_Y) < t \quad \wedge \\ &d(e_X, e_Y) = \min_i d(e_{Xi}, e_Y) = \min_j d(e_X, e_{Yj}) \end{aligned} \tag{5}$$

4 Ranking Consolidated Entities

We continue our previous Example 1 to illustrate our procedure. First, we obtain the result lists RS_X for each source X as depicted in Figure 2a. In addition, we have now a set of edges A_{same} linking co-referent entities. Figure 3 depicts this situation for our example, we see the three result lists and four a_{same} edges (arrows). The co-references A_{same} are either obtained through our consolidation process, are already part of the data or provided by external services such as `http://sameas.org`. We aim at merging these entities into one ranked list while taking the coreferences into account. In the following we present our ranking model for consolidated entities and then show our strategy for exploiting co-references.

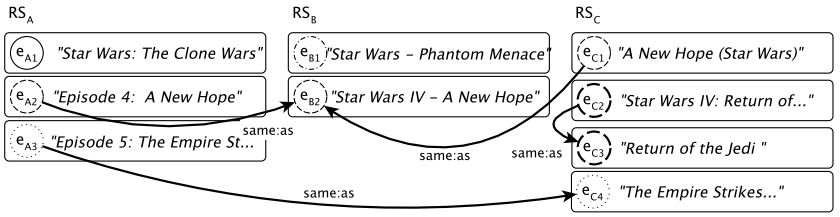


Fig. 3. Three lists RS_X with four $a_{\texttt{same:as}}$ edges (arrows)

4.1 Ranking for Structured Web Data

The general concept we apply for ranking is based on pseudo-relevance feedback [10]. We adapt this idea and apply it to federated entity search. In line with this concept, we build two models, one Query Model (QM) capturing the information need with the help of relevance feedback and Resource Models (RM) representing results to be ranked. Each RM is scored against QM and sorted by its score into the final result list. Both models, QM and RM, share in general the same structure as entities, as described in Section 3.1, i.e. they contain a set of attribute labels A' and a corresponding LM $P \in \mathcal{P}$ for each attribute. Formally, the model $M \in \{QM, RM\}$ is a 3-tuple $M = (\mathcal{E}, A', \mathcal{P})$. We denote the set of entities $\mathcal{E}$ of model M as $\mathcal{E}(M)$ and the set of attributes A' of model M as $A'(M) = \{a | a \in A'(e), \exists e \in \mathcal{E}(M)\}$. As before for the consolidation, we use the JSD (Equation 2) to measure the distance between two corresponding LMs for all attributes that QM and RM have in common:

$$Score(QM||RM) = \sum_{a \in \bigcap_X A'(RS_X)} JSD(P_{QM}(w|a)||P_{RM}(w|a))^{\frac{1}{2}} \tag{6}$$

The LMs of QM and RM, see Equation 7, are computed from the respective LMs of the entities that are comprised by the model and each entity LM P_e (Equation 1) is weighted with an entity specific weight $\mu(e)$:

$$P_M(w|a) = \frac{\sum_{e \in \mathcal{E}(M)} \mu(e) P_e(w|a)}{\sum_{e \in \mathcal{E}(M)} \mu(e)} \tag{7}$$

The weight μ is the crucial part of the query model QM. For RM the weight is constant $\mu = 1$. The weight allows to control the impact of each entity on the query model. With the weight μ, we adapt the ranking framework to the federated search setting and exploit the ranking of the individual sources. We use the discounted rank $r(e_X)$ of the entity e_X in the result list RS_X to weight its influence on the query model:

$$\mu(e_X) = \frac{1}{\log(1 + r(e_X))} \tag{8}$$

By using the ranks in μ, we take advantage of the ranking of the sources. Although the sources do not provide any information explicitly about themselves, all their knowledge, such as domain expertise, popularity, click-data, and

other signals, are incorporated in their ranking function and thereby implicitly conveyed in the ranking. Moreover, we tie the importance of an entity represented by its rank to the content and the structure of the entity, which is captured by the LM of the entity. For QM, we use all entities returned by the sources, i.e. $\mathcal{E}(QM) = \bigcup_X RS_X$ and we construct one RM for each entity. Note that an advantage of the above technique is that it is entirely parameter free. The whole ranking procedure takes all its ingredients from the results returned by the sources for the initial query. This is an important feature for dynamic web environments, where data sources may (dis-)appear frequently and a prior integration into the federated search process is not possible.

4.2 Ranking Consolidation

Given the result lists RS_X, we consider each entity $e \in \bigcup_X RS_X$ individually and construct a corresponding model RM for each entity. Then, we compute a score for each RM and sort each entity by its score to obtain a ranked list of entities. This ranked list contains all entities in $\bigcup_X RS_X$, one entity on each rank as depicted in Figure 2(b). At this point we have a ranked list without taking advantage of the co-references. In the experimental Section 5, we refer to this stage as CRM_w. Now, we allow sets of entities on each rank instead of a single entity. We iterate through the ranked list from the best to the last ranked entity. During this iteration we make use of the co-references A_{same}. If we observe an entity that has co-references, we position the set of all co-referent entities on this rank and remove them from their original ranks. Within each rank, the entity previously ranked highest is first and then we order the co-referent entities by their previous ranks, see Figure 2(c). The result of this strategy is a list of ranked sets. In the next section, we refer to this consolidated ranking strategy as CRM_c.

5 Experiments

We conducted experiments on consolidation and on ranking in two real-world scenarios. In one scenario users search for movies and in the another one for scientific publications. We used publicly accessible APIs available on the Web as sources of entities. Table 1 lists the sources for both scenarios (RT is abbr. for rottentomatoes.com and MS for Microsoft Academic Search). We used Yahoo! Dapper[2] to mimic an API using the site search of Citeseer and ACM. All data used in the experiments is available at `http://www.aifb.kit.edu/web/Dhe/data`.

Real-World Web Search Queries. We extracted 50 real Web search queries for each scenario from a Web search engine query log. For each scenario, we manually created a list of more than ten hostnames, which contains those of the data sources and highly popular sites. We sampled only queries having at least two clicks on one of these hostnames to obtain queries for our scenarios. Details on the query sets and the obtained result lists are given in Table 1 and the first six queries of each set are shown in Table 2.

[2] `www.open.dapper.net`

Table 1. Queries and obtained result lists RS

	Source	#q	$\|q\| \pm \sigma$	max$\|RS\|$	$avg\|RS\| \pm \sigma$	$\|RS\|=\emptyset$
Movie	MovieDb	50	2.58±1.3	20	6.18 ± 7.22	6
	Netflix	50	2.58±1.3	100	81.7 ± 30.4	0
	RT	50	2.58±1.3	50	12.0 ± 15.5	0
Publ.	Arxiv	50	4.4±2.1	100	83.2±36.2	0
	ACM	50	4.4±2.1	20	18.6±4.21	0
	Citeseer	50	4.4±2.1	10	9.2±2.51	3
	MS	50	4.4±2.1	100	89.0±27.3	0

Table 2. Queries with *movie*-related intend (left) and *scientific* intend (right)

mission impossible 4	parameter selection in particle swarm optimization
the debt	mobility models in inter-vehicle communications literature
hobbit	computer effective to academic learning
cowboys and aliens 2011	bivariate f distribution
the hunters 2011	werner krandick
star wars	using truth tables to evaluate arguments

Ground Truth. We obtained the ground truth for both tasks through expert judgments. The distributions of the co-references between the sources are given in Table 3. We can observe that co-references exist not just between but also within the results of a data source. Noteworthy, one source (MS) is dominant in the publication scenario and is part of 83% of the co-references. We followed the methodology of [15] to obtain relevance judgments for the ranking evaluation. We rated the top-10 results for each query. In total, there are 604 relevant entities for the movie scenario, which are distributed among the sources as follows: RT: 40%, Netflix: 36%, MovieDb: 23%. For the publication scenario, the raters judged 997 entities as relevant. The distribution of the relevant results is here highly skewed. MS returned 53% of the relevant results, ACM 24%, Arxiv 15%, and Citeseer 8%. Details on the ground truth are shown in Table 4, such as the number of raters and in particular the inter-rater agreement measured on the "Overlap" with Krippendorff's α for ordinal values [16]. Overall, we consider the agreement of $\alpha_{ordinal} > 0.66$ high enough to rely on the ground truth [17].

Table 3. Ground truth co-references

	MovieDb	RT	Netflix
MovieDb	4		
RT	179	33	
Netflix	162	248	10
		Total:	636

	ACM	Arxiv	Citeseer	MS
ACM	25			
Arxiv	14	62		
Citeseer	13	1	5	
MS	239	75	59	232
			Total:	725

5.1 Consolidation Results

The main focus of our work is on the ranking of consolidated entities. In order to obtain co-references, we applied the procedure described in Section 3. In Figure 4 we see the effect of the threshold t, the only parameter necessary in our approach, on the the metrics *F1*, *Precision*, and *Recall* w.r.t to coreferences between sources. As we have seen in the previous section, many co-references exist also within one data source. We can assume that within a source the same vocabulary is used and therefore entities are inherently closer to each other in terms of our similarity metric compared to entities of different sources. Hence, a lower threshold is needed when consolidating entities of the same source. As a consequence, we reduce t by 0.2 in this case and report evaluation results for $t_{\text{movie}} = 0.7$ and $t_{\text{pub}} = 0.6$ for the respective scenario. We evaluate consolidation from two perspectives. First, we look at each single co-reference link and second, we evaluate the entire co-reference sets created from these links. Each co-reference is classified as true/false positive/negative (abbr. TP, FP, TN,FN). In Table 5c and 5d we see the confusion matrix for both scenarios over the entire query set. The consolidation performance as an average of the co-references created for each query is reported in Table 5a and the corresponding numbers for the sets of co-references are shown in Table 5b. Overall, the performance numbers are high. Although a direct comparison is not possible, the numbers are in the same order of magnitudes as previously reported for supervised consolidation [18]. We now apply these co-references for consolidated retrieval in the next section.

Table 4. Ground truth statistics and agreement α

	Raters	Subjects	Ratings	Overlap	$\alpha_{ordinal}$
Consolidation					
Publications	6	3076	4246	1170	**0.7596**
Movie	3	5783	6061	278	**0.8204**
Relevance					
Publications	6	2736	3022	286	**0.7051**
Movie	3	1616	1992	376	**0.6919**

Table 5. Consolidation performance

Avg. per Q	Movie	Publ.
F1-score	0.8233	0.7672
Accuracy	0.9982	0.9996
Precision	0.8063	0.8636
Recall	0.8781	0.7118

(a) Coreferences

Avg. per Q	Movie	Publ.
#Co-ref Sets	6.9799	9.7000
Set size	2.5100	2.0740
Set Purity	0.7737	0.8713

(b) Sets of coreferences

Movies	
TP:556	FP:166
FN:80	TN:286571

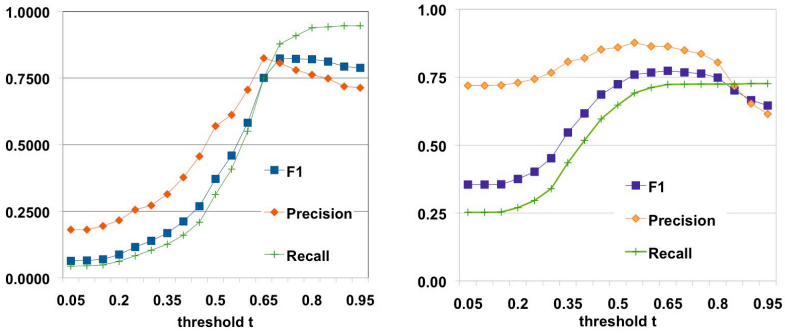
(c) Confusion matrix

Publications	
TP:518	FP:77
FN:207	TN:1049954

(d) Confusion matrix

5.2 Ranking Evaluation

We aim to answer two questions: What is the effect of federated search on retrieval compared to the performance of the sources separately? Does consolidation improve federated search? We assess the ranking using Normalized Discounted Cumulative Gain (*NDCG*) and report the results in Table 6a for the movie and in Table 6b for the publication scenario and indicate statistically significant improvements using Fisher's two-sided, paired randomization test [19].



(a) movie (b) publication

Fig. 4. Consolidation over threshold t

Systems. We implement our *Consolidated Relevance Model (CRM)* approach in two different ways. (1) We employ a federated version ***without*** using co-references (CRM_w) and (2) a federated and ***consolidated*** version exploiting co-references (CRM_c) as described in Section 4. We compare CRM against two baselines, the individual rankings of the sources and the *Multinomial Preference Model (MPM)*, a state-of-the-art rank aggregation strategy [12]. We use an unsupervised version of MPM, i.e. without the supervised adherence parameter, and study two preference encodings. The first encoding $C(e_i, e_j)$ is binary (labeled *MPM*), where one entity e_i is preferred over entity e_j when e_i has a lower rank ($r(e)$ denotes the rank of e in the result list RS):

$$C(e_i, e_j) = \begin{cases} 1 \text{ if } r(e_i) < r(e_j) \\ 0 \text{ otherwise} \end{cases} \tag{9}$$

The second encoding exploits the difference between ranks to express the degree of how much e_i is preferred over e_j using discounted ranks (labeled with subscript d as MPM_d):

$$C_d(e_i, e_j) = \begin{cases} \frac{1}{\log(1+r(e_i))} - \frac{1}{\log(1+r(e_j))} & \text{if } r(e_i) < r(e_j) \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

Evaluation Settings. Note that all systems return a ranked list of individual entities except for those that make use of consolidation, where each result in the

ranked list returned by MPM, MPM_d and CRM_c represents a set of entities instead of a single entity as illustrated in Figure 2(c). In order to assess the relevance of such a set, we use the best ranked entity within that set as the representative element, called the label as depicted in Figure 2(c). The relevance of the set is determined based on the relevance of its label, except for *expand* as described below. We evaluate the systems in three different settings:

Table 6. Retrieval performance (NDCG)

System	Std	NRel	Expand
MovieDb	0.4128	0.4025	n/a
RT	**0.5360**	**0.5165**	n/a
Netflix	0.5191	0.5141	n/a
CRM_w	**0.8699**†	**0.4992**	n/a
MPM	0.4936	0.5037	0.8070
MPM_d	0.5232	0.5232	0.8366
CRM_c	**0.5787**†*	**0.5515**$^{\circ\bullet}$	**0.8744**$^{\circ}$

†stat. diff. $\alpha < 0.05$ to RT, *to MPM, MPM_d
$^{\circ}$stat. diff. $\alpha < 0.05$ to MPM, $^{\bullet}$to CRM_w

(a) Movie scenario results (NDCG)

System	Std	NRel	Expand
Arxiv	0.1824	0.1737	n/a
ACM	**0.3537**	**0.3455**	n/a
Citeseer	0.1630	0.1551	n/a
Publ. 3 sources (Arxiv, ACM, Citeseer)			
CRM_w	**0.3592**	**0.3310**	n/a
MPM	0.2273	0.2273	0.2434
MPM_d	0.2541	0.2542	0.2697
CRM_c	**0.3524***	**0.3462**$^{\star\bullet}$	**0.3697***
Publ. 4 sources (as above and MS)			
MS	**0.6474**	**0.5976**	n/a
CRM_w	**0.5463**	**0.4430**	n/a
MPM	0.4568	0.4230	0.4894
MPM_d	0.4869	0.4743	0.5291
CRM_c	**0.5096**	**0.4822***	**0.5604**

*stat. diff. $\alpha < 0.01$ to CRM_w

(b) Publication scenario results (NDCG)

Std: First, we assess the results in the *standard* way by going through the ranked lists as they are returned by the systems and simply assess the relevance of each rank using the ground truth.

Nrel: In the Std. setting, results are considered relevant even if the same results (i.e. co-referent entities) have been seen in the list before. The *Nrel* setting accounts for redundancy by considering subsequent occurrences of co-referent entities as *non-relevant*, as suggested by [18]. Even if a result is relevant according to the ground truth, it is considered here as not relevant when a co-reference has already been seen.

Expand: The third setting gives special treatment to the systems MPM, MPM_d and CRM_c that perform consolidation. In the previous settings, relevance assessment of these systems is simply based on the labels of result sets. In this *expand* setting, we assess the results in the way proposed for clustered IR [20]. The idea is that a user goes from the top to the bottom of the result list, and checks the label of each cluster (set of results in this case). If a label is considered

relevant, the set is expanded and each entity in the set is assessed individually using the ground truth.

Std Results. First, we look at the effect of federation. When comparing the single sources, shown in the first three lines in Table 6a for the movie scenario, we observe that RT performs best (bold digits). Further, note that the performance differences among these sources are relatively small. The federated approach CRM_w outperforms the individual results by 62%. For the publ. scenario, we look at two setups, one with the three sources that share about the same amount of co-references and relevant results, and a second setup with a fourth source - MS, which is 'an outlier' because it contains 53% of the relevant results and is part of 83% of the co-references, as described above. For the 3-source setup, we observe a different initial situation, see Table 6b. The best source ACM performs about twice as good as the second best source Arxiv. Given this unbalanced situation, our federation approach CRM_w performs only marginally better than the best source. When we look at the publ. scenario with 4 sources in the lower part of this table, we observe that the source MS is again twice as good as the previously best source ACM. Further, adding MS to the pool of sources, improves the performance of CRM_w. However in this skewed setting, the federated approach CRM_w performs not as good as the best source MS, but better than the three other sources. In summary for the std. setting, we observe that federation improves retrieval if the sources have about the same performances. Otherwise, it yields improvements upon most sources but cannot guarantee the best performance. Next, we investigate the effect of federation in combination with consolidation, i.e. the systems MPM, MPM_d and CRM_c. Through consolidation entities are grouped into sets and as a consequence there are less (relevant) results, i.e. (relevant) set labels, in the ranked list after consolidation than entities in the list before consolidation. In the movie case, where more co-reference sets exist (3.6 sets per query in the top10 ranks), we observe as expected that NDCG is much lower than without consolidation. The same holds for the publ. scenario although the difference is smaller because there are fewer and smaller co-reference sets (2.9 sets/query in the top10 ranks). Overall, we observe that federation without consolidation performs best when assessing relevance using the *standard* method. We note that however, since the ranked list with consolidation contains sets of entities, it actually captures much more (relevant) results that are not considered when only assessing their labels.

NRel Results. We perform the same analysis as before, but now regard redundant results as not relevant. Different to the std. setting, we observe that the federated system CRM_w performs worse than the best single data source for both movies and the two publication setups. This indicates there were many co-referent results (redundancy) that are not reflected in the results of the std. setting. In summary, when taking redundancy into account, we observe that federation alone no longer improves over the single data sources. If we investigate the combined effect of federation and consolidation with the system MPM, MPM_d, and CRM_c, we observe a different result. Now, the consolidated CRM_c improves

upon the non-consolidated CRM_w system in all cases. Further, CRM_c outperforms both MPM models, which do not always outperform the non-consolidated run. For the 4-sources publ. scenario, we observe that the consolidated run improves upon the non-consolidated runs, but not upon the outlier source MS. In summary, we observed that consolidation helps federated search when redundant results are considered non-relevant.

Expand Results. Also here CRM_c consistently improves upon the MPM models. To see the effect of expanding relevant sets as opposed to only using their labels (and keeping sets with non-relevant labels collapsed), we compare the results of *expand* with the best results obtained in the std. setting, where federated search without consolidation (CRM_w) performed best. We observe that CRM_c also slightly improves upon CRM_w for both scenarios. This means that consolidated federated search (CRM_c) actually outperforms federated search (CRM_w) when the sets' content representing consolidation results are taken into account. Hence, consolidation can even be useful when redundancy is not considered in the evaluation procedure.

Runtime Performance. The main focus of our work is the effectiveness of ranking strategies. We measured the runtime performance of CRM_c on a standard laptop with Intel Core 2 Duo 2.4 GHz CPU, 4 GB memory and 5400rpm HDD. On average, consolidation took 0.7s for an average of 117 entities per query in the movie scenario, and 2.2s for 206 entities in the publication scenario. Ranking took 0.4s for the movie scenario and 1.4s for the publication scenario. These run times are small compared to the amount of time necessary for remote API calls, which took 4s for the publication case and 31s for the movie case. In the latter case, the time includes several API calls because some movie sources return a list of IDs for a query and then each ID has to be fetched individually. This is because these APIs were in fact designed for a different use case (browsing) and are thus, not suitable for online search. In addition, we used developer keys, which may have a lower priority than production keys when requesting data. A demonstrator of our system is available online at `http://km.aifb.kit.edu/services/conesearch`.

6 Related Work

Entity Consolidation is also referred to as record linkage, instance matching or object de-duplication, has a long history in database research and many approaches have been proposed [5–7]. Note, that consolidation is different to fusion, where the goal is to blend instances into one object [21]. With respect to our work, we focus on entity consolidation in a Web context with limited data access. Recent work on consolidating entities for Web search shows that consolidation improves search performance by achieving more diverse and less redundant results [18]. While they use a *supervised* approach relying on training examples, we propose an *unsupervised* approach that is more suitable to uncooperative settings where training data is not available. In a Semantic Web context, the

task of entity consolidation is equivalent to establishing `owl:sameAs` links between entities. Using statistics derived from the entire datasets to establish such links as been studied by [22], who present a self-learning approach, by [23], who focus on scalability, and by [24], who propose statistics as well as logic-based approaches to match entities. However, all these unsupervised approaches are not targeting at search, but have the goal to integrate entire datasets. Hence, they require access to the full datasets, while our approach only uses data retrieved for a given search query. The aspect of *query-time* data integration has been studied for schema alignment during Web data search [11], to identify and consolidate records helpful to process a database query [25], and in the context of probabilistic databases [26].

Entity Search has been studied in many approaches [1, 27–30]. The aforementioned approaches assume a central index comprising the entire data collection. Integrating the data of several sources has been studied in vertical search [31], where the results of different verticals are combined at the front-end level but not at the level of the search algorithms.

Federated Search (distributed IR) has been thoroughly studied for document retrieval [8, 9, 32], where the unit of retrieval are textual documents and not structured entities. *Query translation* for federated entity search has been investigated by [30], who use source-specific query generators to adapt a structured query to each source. *Source selection* and ranking algorithms are studied in [27]. We use the *rank aggregation* strategy of [12] as baseline in our experiment. It requires the presents of coreferences to form a consensus ranking. Our work differs by consolidating entities at query-time and incorporating content, structure and the original rank into the ranking strategy.

7 Conclusion

We have presented the first unsupervised solution for federated entity search using on-the-fly consolidation for uncooperative environments. Our consolidation as well as our ranking technique are incorporated into the language model based IR framework and operate without prior knowledge or training examples, but only on the data obtained for one query and hence are suitable for search over Web data sources with access through APIs. Our experiments investigate the effects of consolidation and federation on retrieval performance. The results show that our approach outperforms a state-of-the-art preference aggregation strategy and that consolidation improves the retrieval performance.

Acknowledgements. This work was partially supported by the EU FP7 project XLIKE (Grant 288342).

References

1. Balog, K., Carmel, D., de Vries, A.P., Herzig, D.M., Mika, P., Roitman, H., Schenkel, R., Serdyukov, P., Tran Duc, T. (eds.): Proc. 1st Int. Workshop on Entity-Oriented and Semantic Search. JIWES, SIGIR (2012)
2. Guo, J., Xu, G., Cheng, X., Li, H.: Named entity recognition in query. In: SIGIR, pp. 267–274 (2009)
3. Pound, J., Mika, P., Zaragoza, H.: Ad-hoc object retrieval in the web of data. In: WWW, pp. 771–780 (2010)
4. Wick, M.L., Singh, S., McCallum, A.: A discriminative hierarchical model for fast coreference at large scale. ACL (1), 379–388 (2012)
5. Doan, A., Halevy, A.Y.: Semantic integration research in the database community: A brief survey. AI Magazine 26(1), 83–94 (2005)
6. Elmagarmid, A.K., Ipeirotis, P.G., Verykios, V.S.: Duplicate record detection: A survey. IEEE Trans. Knowl. Data Eng. 19(1), 1–16 (2007)
7. Köpcke, H., Rahm, E.: Frameworks for entity matching: A comparison. Data Knowl. Eng. 69(2), 197–210 (2010)
8. Callan, J.: Distributed information retrieval. In: Croft, W. (ed.) Advances in Information Retrieval. The Inf. Retrieval Series, vol. 7, pp. 127–150. Springer (2000)
9. Shokouhi, M., Si, L.: Federated search. Foundations and Trends in Information Retrieval 5(1), 1–102 (2011)
10. Lavrenko, V.: A generative theory of relevance. Springer, Berlin (2009)
11. Herzig, D.M., Tran, T.: Heterogeneous web data search using relevance-based on the fly data integration. In: WWW, pp. 141–150 (2012)
12. Volkovs, M., Zemel, R.S.: A flexible generative model for preference aggregation. In: WWW, pp. 479–488 (2012)
13. Chaudhuri, S., Chen, B.C., Ganti, V., Kaushik, R.: Example-driven design of efficient record matching queries. In: VLDB, pp. 327–338 (2007)
14. Neumayer, R., Balog, K., Nørvåg, K.: On the modeling of entities for ad-hoc entity search in the web of data. In: Baeza-Yates, R., de Vries, A.P., Zaragoza, H., Cambazoglu, B.B., Murdock, V., Lempel, R., Silvestri, F. (eds.) ECIR 2012. LNCS, vol. 7224, pp. 133–145. Springer, Heidelberg (2012)
15. Blanco, R., Halpin, H., Herzig, D.M., Mika, P., Pound, J., Thompson, H.S., Tran, D.T.: Repeatable and reliable search system evaluation using crowdsourcing. In: SIGIR, pp. 923–932 (2011)
16. Hayes, A.F., Krippendorff, K.: Answering the call for a standard reliability measure for coding data. Communication Methods and Measures 1(1), 77–89 (2007)
17. Krippendorff, K.: Reliability in content analysis. Human Communication Research 30(3), 411–433 (2004)
18. Dalton, J., Blanco, R., Mika, P.: Coreference aware web object retrieval. In: CIKM, pp. 211–220 (2011)
19. Smucker, M.D., Allan, J., Carterette, B.: A comparison of statistical significance tests for information retrieval evaluation. In: CIKM, pp. 623–632 (2007)
20. Leuski, A.: Evaluating document clustering for interactive information retrieval. In: CIKM, pp. 33–40 (2001)
21. Rahm, E., Thor, A., Aumueller, D., Do, H.H., Golovin, N., Kirsten, T.: ifuice - information fusion utilizing instance correspondences and peer mappings. In: WebDB, pp. 7–12 (2005)
22. Hu, W., Chen, J., Qu, Y.: A self-training approach for resolving object coreference on the semantic web. In: WWW, pp. 87–96 (2011)

23. Song, D., Heflin, J.: Automatically generating data linkages using a domain-independent candidate selection approach. In: Aroyo, L., Welty, C., Alani, H., Taylor, J., Bernstein, A., Kagal, L., Noy, N., Blomqvist, E. (eds.) ISWC 2011, Part I. LNCS, vol. 7031, pp. 649–664. Springer, Heidelberg (2011)
24. Hogan, A., Zimmermann, A., Umbrich, J., Polleres, A., Decker, S.: Scalable and distributed methods for entity matching, consolidation and disambiguation over linked data corpora. J. Web Sem. 10, 76–110 (2012)
25. Bhattacharya, I., Getoor, L.: Query-time entity resolution. J. Artif. Intell. Res. (JAIR) 30, 621–657 (2007)
26. Ioannou, E., Nejdl, W., Niederée, C., Velegrakis, Y.: On-the-fly entity-aware query processing in the presence of linkage. PVLDB 3(1), 429–438 (2010)
27. Balog, K., Neumayer, R., Nørvåg, K.: Collection ranking and selection for federated entity search. In: Calderón-Benavides, L., González-Caro, C., Chávez, E., Ziviani, N. (eds.) SPIRE 2012. LNCS, vol. 7608, pp. 73–85. Springer, Heidelberg (2012)
28. Blanco, R., Mika, P., Vigna, S.: Effective and efficient entity search in rdf data. In: Aroyo, L., Welty, C., Alani, H., Taylor, J., Bernstein, A., Kagal, L., Noy, N., Blomqvist, E. (eds.) ISWC 2011, Part I. LNCS, vol. 7031, pp. 83–97. Springer, Heidelberg (2011)
29. Cheng, T., Yan, X., Chang, K.C.C.: Entityrank: Searching entities directly and holistically. In: VLDB, pp. 387–398 (2007)
30. Endrullis, S., Thor, A., Rahm, E.: Entity search strategies for mashup applications. In: ICDE, pp. 66–77 (2012)
31. Arguello, J., Diaz, F., Callan, J.: Learning to aggregate vertical results into web search results. In: CIKM, pp. 201–210 (2011)
32. Nguyen, D., Demeester, T., Trieschnigg, D., Hiemstra, D.: Federated search in the wild: the combined power of over a hundred search engines. In: CIKM, pp. 1874–1878 (2012)

ProSWIP: Property-Based Data Access for Semantic Web Interactive Programming

Silviu Homoceanu, Philipp Wille, and Wolf-Tilo Balke

Institute for Information Systems,
Technische Universität Braunschweig, Germany

Abstract. The Semantic Web has matured from a mere theoretical vision to a variety of ready-to-use linked open data sources currently available on the Web. Still, with respect to application development, the Web community is just starting to develop new paradigms in which data as the main driver of applications is promoted to first class status. Relying on properties of resources as an indicator for the type, property-based typing is such a paradigm. In this paper, we inspect the feasibility of property-based typing for accessing data from the linked open data cloud. Problems in terms of transparency and quality of the selected data were noticeable. To alleviate these problems, we developed an iterative approach that builds on human feedback.

1 Introduction

The amount of data available on the Web has considerably increased in the last few years. Despite huge efforts in the area of the Semantic Web to make such web data machine-processable, only a few applications have been developed that can take full advantage of this data. Besides the general sparseness of semantic data, this behavior is currently explained by the different representation formalisms of semantic data and application programming languages causing a problem of data-model/programming language interoperability. Most semantic data stores provide data in Resource Description Framework (RDF) graphs or ontologies represented in the OWL Web Ontology Language. Accessing such data from state-of-the-art object oriented (OO) programming languages requires mappings from entities and ontology categories to programming structures like classes. Fortunately, similarly to Object Relational Mapping (ORM) [1] for relational databases, there are frameworks available that map RDF and/or OWL to programming structures by means of textual code generation. Well known frameworks include Jena [2] and RDFReactor [3].

However, generated code is often unintelligible, hard to customize and almost impossible to maintain. While some frameworks make customization and maintainability more convenient by including support for IDEs, compile-time meta-programming [4] represents a better technique to cope with the interoperability problem. With compile-time meta-programming, developers can programmatically generate required classes instead of providing them directly into the source code. One such approach was recently presented by Microsoft as a feature of F# 3.0 (http://msdn.microsoft.com

H. Alani et al. (Eds.): ISWC 2013, Part I, LNCS 8218, pp. 184–199, 2013.

/en-us/library/hh156509.aspx). Called type provider, it's a component providing types, properties and methods for an external data source without having to write these types manually for each application. Type providers seem promising for accessing data from single data stores. But because each data source may have its own vocabulary, an RDFS type provider for the Linked Open Data cloud (LOD) would be useless without proper cleansing. With no global ontology to drive the cleansing and alternative solutions like automatic ontology alignment offering just average quality, such a type provider would require manual mapping during application development.

When writing an application, software engineers have some mental representation of "things" that are required for the application. It is common knowledge in cognitive psychology (imported in information science [5]) that concepts take the place of thoughts. They are represented through symbols (words, sounds, etc.), defined intensionally by a set of properties, and extensionally by a set of entities. The goal in programming with web data is to easily access the entities that correspond to a concept the software engineer thinks of. This concept may easily be expressed by its symbol, a word label. In the LOD cloud, entities are associated with concept labels by means of the rdf:type property. Detrimental to our purpose, types are provided in different granularities, e.g., Movie, Animation, FrenchFilm, etc. We found about 1,700 entity types for movies in the LOD cloud. Furthermore, for some entities, no type is provided. In consequence, accessing entities through their type labels is difficult.

We believe that a property-based data access model as recently sketched in [6, 7] is more suitable for programming with semantic data. The type information for such a programming approach is given by properties: A type is defined by a set of required properties, and every entity with at least those properties is part of that type. When designing an application, during the data modeling phase, developers usually think in terms of entities – no clear cut types, but concepts like Movie, Actor, etc. When writing code, these concepts are bound to properties which are required for the program logic. This way, concepts are extended to a minimal intensional definition comprising core properties (e.g. Movie ≡ {Title, Genre, Director}) required by the program. Similar to the case of structural subtyping or DuckTyping [8], this definition is used to identify entities that belong to the concept (in this case all entities providing values for Title, Genre, Director are considered to be Movies). We inspected the feasibility of such a programming paradigm for accessing data from the LOD cloud. Our experiments show that simple property-based data access can lead to selecting all kinds of entities. For example Music, Video-Games, and Books were also selected when trying to access Movies. The quality of the selected data is poor if properties describing the intended concept are not well chosen. Based on this observation, we propose ProSWIP (Property-based Semantic Web Interactive Programming), an approach which empowers property-based data access while maintaining quality under control. Part of a cloud-based centralized service for programming the Semantic Web, ProSWIP will be accessible from IDEs by means of plugins. Starting from properties provided by application developers, ProSWIP estimates the quality of the selected data and if necessary, identifies additional properties that have high positive impact on the quality. In an iterative process, it assists developers to extend the property-based type definitions while checking that the extended definition still matches their intentions.

The contribution of this paper can be summarized as follows: An extensive inspection of the property-based paradigm's feasibility for accessing data from the LOD cloud; the presentation and evaluation of a quality metric enabling transparency for this programming paradigm; and the presentation and evaluation of a property selection method for better data quality.

2 Property-Based Data Access - Use Case

To assess the feasibility of the property-based paradigm for accessing data from the Web, we conducted an experiment focused on developing applications related to movies: When writing such applications developers rely on variables that represent movie properties. These properties are used in the property-based paradigm as filters so that all entities from the LOD cloud which have values for those properties are considered to represent movies. By inspecting the selected entities to identify those that actually are movies, we get an impression of the quality of the property-based paradigm. But first, what are the properties developers require for programming applications concerning movies? We conducted an extensive analysis (involving about 6% of the pages on the Web) to find properties typically associated with movies.

Motivated by the improved Web visibility promised by rich snippets, application developers started to adopt the vocabulary provided by schema.org to semantically annotate data published on the Web. Schema.org was launched in 2011 as joint initiative of major search engine providers like Bing, Google, Yahoo and Yandex to provide a unified set of vocabularies which web masters and application developers can use to semantically annotate data published on the Web. The goal of the project was to ultimately empower semantic Web search. Currently, schema.org provides a collection of 406 hierarchically built schemata for various concepts ranging from organizations, persons and events to creative works like movies, music or books. On average, schemata comprise about 34 attributes representing properties of the corresponding concepts. Movie (http://schema.org/Movie), with a total of 62 attributes, belongs to the fewer schemata that are described in more detail. While any subset of these properties can theoretically be used by application developers to refer to the Movie concept, some properties may be preferred: To establish which of the 62 properties are mostly being used when referring to movies, we analyzed a crawl of 870 million websites. Known as ClueWeb12 (http://boston.lti.cs.cmu.edu/clueweb12/) this Web crawl is publicly available as a corpus and consists of only English language sites, which have been crawled between February and May 2012. More than a year after schema.org was introduced, only about 1.56% of the web sites from ClueWeb12 comprised data that was annotated with schema.org. Overall, only 192 schemata out of 406 from schema.org were used for annotating data. On average, annotations comprised 4.6 properties. For movies, we observed about 40,000 annotations. In Table 1 we present a list of the properties most frequently used for annotating movie data. For movies, annotations comprised on average 4.5 properties, with a minimum of 1 and a maximum of 13 properties. These numbers are rather low considering that the Movie schema comprises 62 properties. This observation is not particular to movies but has

Table 1. Top "Movie" properties (with frequency above 30%) from schema.org frequently used for annotating movie data on Web pages from the ClueWeb12 corpus

Property	Movies annotated with property
Title	78 %
Description	56 %
URL	44 %
Director	39 %
Genre	38 %
Actors	38 %
AggregateRating	33 %

been made for other schemata like events or organizations as well, indicating that the property-based approach may suffer from under-specification.

Assuming that, in part, annotated data published on the Web surfaced as a result of some Web application, most developers require on average 3 to 5 properties. These properties are most likely the ones that have frequently been annotated. For the Movie concept, the most probable property-based definitions are {Title, Description, URL}, {Title, Description, URL, Director}, etc.

According to the property-based paradigm, all entities from the LOD cloud fulfilling these properties represent movies. We rely on the Billion Triples Challenge 2012 (BTC) dataset to represent the LOD cloud. BTC comprises about 1.4 billion quads of the form (subject, predicate, object, source) crawled from major LOD data stores like Datahub, DBpedia, Freebase, and others during May and June 2012. Entities and properties are provided as unique identifiers (URIs) in the quad subjects and predicates respectively. Sources are not relevant for our approach and will be ignored in this paper. The process of selecting data for a set of properties provided in natural language works as follows: (i) Property URIs are identified for each property. For this purpose, all subjects from tuples of the form (*, rdfs:label, *p*) are selected for each property *p* (* is a wildcard that may be substituted by any URI). Synonym sets provided by WordNet or obtained through the owl:sameAs predicate are used to extend the coverage of each property (more details in Section 3.1). (ii) With *p'* as the URI of each property *p*, the entities to be selected are the set of all distinct subjects *s* for which there are tuples of the form (*s*, *p'*, *) in the BTC dataset (* is a wildcard that may be substituted by any URI or literal). An overview of the selectivity for different property sets is provided in Table 2(a). While Title, Description and URL are quite general (1,5 million entities), Director, Actors and especially Genre significantly reduce the number of relevant entities.

Precision and recall are the standard measures for evaluating the quality of retrieved information or, in our case, the quality of selected entities. Precision is for our scenario defined as the proportion of entities representing movies out of all selected entities, while recall is defined as the proportion of selected movies out of all movies present in the BTC dataset. Computing precision and recall is not trivial in this case since it requires recognizing entities that are movies. As the rdf:type property connects

entities to different types that may be related to movies (e.g., Films, Animations, FrenchFilms etc.), such types are difficult to automatically map to the Movie type without a general movie taxonomy. Without claiming full completeness for this experiment, we extracted from the BTC data set a list of movie types by bootstrapping on a seed of movies from the Linked Movie Data Base (LMDB - linkedmdb.org/) and manually inspecting the resulting types. More details about this process are presented in Section 4. In total, we found 1,736 types expressing different kinds of movies. This surprisingly large number is mostly due to the very fine classification provided by YAGO. With these types we identified a total of 87,273 movies in the BTC dataset.

As shown in Table 2(b), the choice of properties has notable impact on the quality of the selected entities: Precision increases from a mere 0.02 to 0.78 by adding one single property to the definition of Movie. Precision values of 0.92 are possible if the "right" properties are chosen. Recall is, with 0.3 for the first three most frequent properties, quite low. The main reason is the sparseness of the data. This becomes extreme in the case of Genre with just a few movies having this property.

Table 2. Nr. of entities from the BTC data set fulfilling each property set (a). The corresponding precision and recall values (b).

Property Set	(a) Nr. of Entities from BTC	(b) Precision / Recall
{Title, Description, URL}	1,447,813	0.018/0.3
{Title, Description, URL, Director}	29,328	0.78/0.26
{Title, Description, URL, Director, Genre}	2,266	0.35/0.01
{Title, Description, URL, Director, Actors}	21,531	0.92/0.23

Overall, the property-based paradigm can lead to high quality/high precision entity selection if properties are well chosen. A major obstacle in the process is the lack of transparency: The application developer has no idea about the quality of the selected entities. Properties belonging to the concept definition are mandatory and values for these properties are required by the application. In consequence, none of the entities missing on any of these properties can be used. But this has a high impact on recall. Combined with the sparse nature of LOD, the more elaborate the definition, the smaller the number of selected entities. In this paper we focus on improving the quality of the selection throughout precision first, by extending the concept definition with a set of well chosen properties. We believe once high quality properties are found, we can tackle the recall problem by building on structural similarity focused on the extending properties, but leave this as the subject of future work.

3 System Description

Starting from a property-based type definition with properties expressed in natural language and a large collection of data representing facts from the LOD cloud, ProSWIP helps the user to keep data quality problems under control: Relying on a

measure of property-based data homogeneity, it measures the quality of the entities that fulfill the property-based definition. If the quality is low, key properties contributing the most to better data quality are found. The user has to finally decide if those properties are part of the type or not. The definition of the intended type is extended to include the user feedback and the process is repeated until the quality reaches a satisfactory level. For this purpose, the following functionality is required: i) identify and select those entities that fulfill the property-based type definition, ii) compute the quality of a collection of entities, iii) find properties that, if added to the set of properties defining the type, significantly improve the quality of the selected data.

3.1 Property-Based Data Access

According to the property-based paradigm, the system selects all entities from the LOD cloud having all properties from a given set. But in the LOD cloud, properties are represented through URIs. Hence, a mapping between the properties in natural language and the URIs is necessary. For this mapping, we rely on the rdfs:label property, an instance of rdf:property providing a human-readable name for a resource. For better coverage, each property is automatically extended beforehand with a list of synonyms from WordNet.

Definition 1 (mapping): Given a property $p \in$ Properties, P_{SYN_p} its set of synonyms from WordNet (including p) and LOD a large set of 3-tuples of the form (subject, predicate, object), we define *map* as a function *map* : Properties $\rightarrow \wp$(URIs) with:

$$p \mapsto \{s | \exists p_i \in P_{SYN_p} : (s, \text{rdfs:label}, p_i) \in LOD\} \tag{1}$$

For some entities the rdfs:label property may be missing. Furthermore, the same property may be present in different data stores under different URIs, possibly connected to each other through the owl:sameAs property. In consequence, in a dictionary-like fashion, each property is actually mapped to a set of URIs all considered synonyms.

Mapping Expansion Algorithm:
With $\Delta_{p,1} \coloneqq map(p)$ define

$$\Delta_{p,i+1} \coloneqq \{s_j' | \exists s_j \in \Delta_{p,i} : (s_j, \text{owl:sameAs}, s_j') \in LOD \vee (s_j', \text{owl:sameAs}, s_j) \in LOD\} \tag{2}$$

$$Dictionary(p) \coloneqq \bigcup_{i=1}^{\infty} \Delta_{p,i} \tag{3}$$

By repeatedly linking elements through synonyms, two or more properties from the definition set may end up being represented by the same set of URIs. This doesn't play any role in the process of selecting the appropriate entities but may surprise the user when accessing values for these properties. Such cases are reported to the user.

At the very core of the property-based paradigm, an entity is relevant with respect to a specific property if there is a statement or fact asserting that the entity has this property. In the context of linked open data, we define the binary relevance of an entity w.r.t. a property as a *hit* function:

Definition 2 (hit): Given some entity $e \in E$ represented by its URI, a property in natural language $p \in$ Properties and *LOD* defined as above, we define *hit* as a function *hit* : (URIs × Properties) → {0, 1} with:

$$hit(e,p) = \begin{cases} 1 & \text{iff } \exists\, p' \in Dictionary(p)\colon (e,p',*) \in LOD \\ 0 & otherwise \end{cases} \tag{4}$$

where * is a wildcard that may be substituted by any literal or URI.

According to the *semiotic triangle* from cognitive psychology [5], concepts are defined intensionally by a set of properties, and extensionally by a set of entities. Aiming for simple yet effective access to entities corresponding to a certain concept we define conceptual variable *types* in the sense of programming, as a set of properties that intensionally define a concept. This type definition may iteratively evolve based on user feedback. Because the user feedback may be negative w.r.t. to some properties (by negative we mean properties that all entities corresponding to the concept definitely shouldn't possess), we define a type as follows:

Definition 3 (type): Given a concept *c*, extensionally defined through the set of entities given by their URIs, E_c, we define the *type* of concept *c* denoted T_c as the set of properties $T_c = P_{c_+} \cup P_{c_-}$ with P_{c_+} the set of positive properties and P_{c_-} the set of negative properties ($P_{c_+} \cap P_{c_-} = \emptyset$), such that:

$$\begin{aligned} &(i) \quad \forall e \in E_c, p \in P_{c_+}\colon hit(e,p) = 1 \\ &\text{(ii)} \quad \forall e \in E_c, p \in P_{c_-}\colon hit(e,p) = 0 \\ &(iii) \quad \forall e \notin E_c\ \exists p \in P_{c_+}\colon hit(e,p) = 0 \end{aligned} \tag{5}$$

While here all properties (initial as well as positive and negative extensions) are treated equivalently, the fact that not all properties extending the definition are required is a starting point for future work. As in the case of properties, in the LOD cloud the same entities may end up having multiple URIs. For the sake of simplicity, we refer to one entity as being uniquely identified by an URI.

More often than not, the number of properties employed to refer to some type of entity is much smaller than the number of properties that would completely define the entity type or intended concept. Actually, extensive experiments presented in Section 2 show that on average only 4.6 (out of an average of 34 existing) properties have been used to link entities to concepts. This suggests that the developer provides a sub-set of properties meant to represent the intended (to us hidden) type. This set of properties is one of the many possible super-types of the intended type. Starting from a property set that builds a type or a super-type for some concept, all entities having all these properties are selected as being relevant for the type or super-type:

Definition 4 (property-based data access): Given a set of properties $T_c = P_{c_+} \cup P_{c_-}$ representing either a type or super-type for a concept c as before, the set of entities selected according to the property-based data access paradigm E_c is the set of entity URIs that fulfill all properties from T_c:

$$E_c = \bigcap_{j=1}^{|T_c|} E_j \tag{6}$$

where $E_j = \{e | hit(e, p_j) = 1 \; if \; p_j \in P_{c_+} \wedge hit(e, p_j) = 0 \; if \; p_j \in P_{c_-}\}$

In the ideal case, for a concept c the set of properties T_c is the *type* of c (not a super-type). Then, the set of selected entities E_c also extensionally defines concept c and should perfectly satisfy the user needs. However, there is a high probability that a super-type is provided. Since the type intended by the application developer is hidden to the system and entities have no clear types, there is no trivial way for checking if the selected entities correspond to the intended concept. The developer also has no feedback whatsoever regarding how good the selected entities match the intended use. This has grave effects on the applicability of the property-based data access paradigm. Aiming for better transparency of the whole approach, in the next section we introduce a measure of quality for the selected entities.

3.2 Quality of the Selected Entities

We measure the quality of entities selected through the property-based model as a function of entity homogeneity. The basic assumption is that the application developer describes simple concepts (like "Movies" or "Books") with all corresponding entities having the same or almost the same properties and not ad-hoc or composed concepts (like "all things having a geo-location"). Consider for example that the developer provides three properties: Title, Description and Genre. Based on these properties a set of eight entities is selected. Besides the three properties, each entity is described by other additional properties like in Table 3. Properties p_4, p_5 and p_6 may be, for instance, Duration, Actors and Director while p_7, p_8 and p_9 could represent ISBN, Pages and Editor. As you may have intuited, entities e_1, e_2, e_3 and e_4 represent movies while the remaining entities represent books. Properties in the LOD cloud may be missing. This is reflected also in this artificial example with movies e_1, e_3 and e_4 providing no values for properties p_4 and respectively p_6. Analogously, for the entities representing books. The rest of the missing values are attributed to the fact that properties p_4, p_5 and p_6 are proper to movies while p_7, p_8 and p_9 are proper to books.

More generally, starting from the set of properties, the system selects a set of entities as described in the previous section. In a relational sense, together with the union of all their corresponding properties (stop properties like rdfs:label, owl:sameAs, rdf:type, etc. are first removed) these entities form a relational schema (as in Table 3). Especially in the field of schema extraction and discovery, the number of null values has successfully been used for establishing the quality of the schema [9] – the better the schema, the fewer null values, the more homogeneous the data. Thus, if the data is homogeneous in terms of structure - their properties - these properties intensionally define a single concept. As a measure of homogeneity we measure the property-based

Table 3. On rows - the entities that are selected for the properties set $\{p_1, p_2, p_3\}$. On columns - all properties that describe any of the selected entities.

	p_1	p_2	p_3	p_4	p_5	p_6	p_7	p_8	p_9
e_1	✓	✓	✓	×	✓	✓	×	×	×
e_2	✓	✓	✓	✓	✓	✓	×	×	×
e_3	✓	✓	✓	✓	✓	×	×	×	×
e_4	✓	✓	✓	✓	✓	×	×	×	×
e_5	✓	✓	✓	×	×	×	✓	✓	✓
e_6	✓	✓	✓	×	×	×	✓	×	✓
e_7	✓	✓	✓	×	×	×	✓	×	✓
e_8	✓	✓	✓	×	×	×	✓	✓	×

similarity between all entities. But there is a problem: Entities may be selected from different data sources (DBpedia, LMDB, etc.). Entities with the same type and from the same source tend to share the same properties, usually due to the focus of each data store. Different sources have different sizes, and small data sources with many properties can introduce null values. These null values are artificially amplified by the size of the data source. To handle this problem, we reduce all entities having the exact same properties to just one *witness*. This way, for the example presented in Table 3, e_3 and e_4 are both represented by one witness: $w_{e_3e_4}$ having the same properties as e_3 or e_4. The same for e_6 and e_7. The rest are their own witnesses. Based on this observation we define the quality of a set of entities as follows:

Definition 5 (quality): With the notations of T_c and E_c as above and W_c as the set of witnesses represented by URIs of entities from E_c, the quality of the selected entities is a function, $Q : \wp(\text{URIs}) \rightarrow [0, 1]$ with:

$$Q(E_c) = \frac{1}{C_2^n} \cdot \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} Sim(w_i, w_j) \tag{7}$$

$\forall w_i, w_j \in W_c, n=|W_c|$ and $sim(w_i, w_j) = \frac{|P_{w_i} \cap P_{w_j}|}{|P_{w_i} \cup P_{w_j}|}$ is the Jaccard similarity index [10]. P_{w_i} is the set of properties of w_i and P_{w_j} is the set of properties of w_j.

While the Jaccard index is most suitable for measuring structural similarity between entities, any other similarity measure may be used here.

For the example introduced in Table 3, the quality of the selected entities is 0.55. If additional information were provided, like the concept the application developer has in mind also has property p_5, or doesn't have property p_7, the entities selected by the property based model restrict to movies only (entities 1 to 4). The quality in this case increases to 0.78, the result being slightly affected by the noise (missing values) in the data. In the following subsection we present how to find properties better separating various types of entities in the result set.

3.3 Property Selection

Finding the list of properties best distinguishing different types is similar to the problem of induction of an optimal decision tree in data classification, which is a hard task. It has been shown that finding a minimal decision tree consistent with the set of labeled entities provided as data is NP-hard [11]. Consequently, greedy algorithms like the C4.5 are applied for solving this problem [12]. When it comes to selecting some property that better discriminates between different types of entities, *information gain* from the field of information theory is the standard measure for deciding the relevance of a property [13]. Generally speaking, the information gain is the change in information entropy from a prior state to a state that takes some information as given. Computing this entropy change is only possible for entities that have class labels (entity types) attached. Types are provided in the LOD cloud by means of the rdf:type property, however entities may have multiple types partly with different granularities e.g., the movie "Gangs of NewYork" has types owl:Thing, schema.org/CreativeWork, dbpedia-owl:Film, yago:VictorianEraFilms and 15 other types. For other movies, types owl:Thing, or schema.org/CreativeWork are missing. All these types are obviously related to each other but without an upper ontology or global type hierarchy, it's difficult to make use of the type property to compute the information gain.

But the type information strongly correlates with the entity properties [14]: In the example presented in Table 3, it's obvious that entities having properties Duration, Actors and Director on top of Title, Description and Genre are movies while entities having ISBN, Pages and Editor are books. The type information is latent in the properties. But the missing values for some entities, as well as the heterogeneity of data sources make it difficult to fold all movies together to just one witness – a property set representing the movie type. Actually what happens is that more witnesses, with more or less similar properties, exist for a single type. The problem of reducing similar witnesses to a dominant type is similar to the problem of dimension reduction.

Principal component analysis (PCA) is the best, in the mean-square error sense, linear dimension reduction technique [15]. In essence, PCA is a basis transformation that seeks to reduce the dimensionality of the data by finding a few orthogonal linear combinations (called principal components) of the original variables capturing the largest variance. Given E_c the set of entities selected according to the property-based data access paradigm, and W_c the set of witnesses of entities from E_c, let X be a $n \times p$ matrix, where n and p are the number of entity witnesses and the number of properties of all witnesses, respectively. Let the matrix decomposition of X be

$$X = UDV^T \tag{8}$$

$Y=UD$ are the principal components (PCs), where the $p \times p$ matrix U is the matrix of eigenvectors of the covariance matrix XX^T, matrix D is a $p \times n$ rectangular diagonal matrix of nonnegative real numbers on the diagonal with customary descending order, and the $n \times n$ matrix V is the matrix of eigenvectors of X^TX. The columns of V are called loadings of the corresponding principal components. Usually the first PCs (capturing the highest data variance) are chosen to represent the dominant dimensions.

For the example introduced in Table 3 (first all data is reduced to binary values and centered on the columns such that the mean of each column is equal to 0), the first PC shows the strongest variance of 1.16. The next two components show a variance of 0.2 and the rest are 0 or close to 0. With respect to the properties, the coefficients of the first PC are clustered together according to their variance (Table 4). For this example, the three property clusters that build on the most significant PC show the existence of two dominant types that differentiate in terms of properties p_4, p_5, p_6 and p_7, p_8, p_9. Showing no variance, properties p_1, p_2 and p_3 can be ignored since they belong to both dominant types.

Table 4. Property coefficients of the first three PCs

Props. \ PCs	PC_1	PC_2	PC_3
p_1	0.00	0.00	0.00
p_2	0.00	0.00	0.00
p_3	0.00	0.00	0.00
p_4	0.35	-0.71	0.00
p_5	0.50	0.00	0.00
p_6	0.35	0.71	0.00
p_7	-0.50	0.00	0.00
p_8	-0.35	0.00	0.71
p_9	-0.35	0.00	-0.71

In general, depending on the selected entity set, more PCs may be significant. To dynamically establish which of them show significant variance, we rely on the ISODATA algorithm, an automatic thresholding approach [16] that identifies thresholds in one dimensional spaces that best separate a set of data points. With the PCs that show variances above the threshold, one dimensional clusterings (agglomerative hierarchical clustering with average inter-cluster similarity) on the coefficients are built for each PC. This way each property is assigned to one cluster for each significant PC. Each set of properties belonging to the same clusters on all significant PCs are grouped together and represent abstract dominant types we will further refer to as *latent types*. For the example in Table 4, considering that only PC_1 is significant, the extracted latent types are $t' \equiv \{p_1, p_2, p_3, p_4, p_5, p_6\}$ and $t'' \equiv \{p_1, p_2, p_3, p_7, p_8, p_9\}$. With these types we can now label entities according to the property-based model. This way, e_2 will be labeled with t' and e_5 with t''. For the future, we plan to introduce a probabilistic approach to increase the labeling recall, but for now all entities missing some values are ignored in the typing process. In this manner a set of labeled entities is created. Entities that fulfill properties for multiple types (Audiobooks in the context of our example) are automatically associated with multiple labels.

With the set of labeled entities, the information gain for a property can be computed as follows:

Definition 6 (information gain): With the notations of T_c and E_c as previously defined and P_U the set of all properties of all entities from E_c, the *information gain* of a property $p \in P_U - T_c$ w.r.t. the entity selection E_c is:

$$Gain(p, E_c) = H(E_c) - \sum_{v \in \{0,1\}} \frac{|E_c|p^v|}{|E_c|} \cdot H(E_c|p^v) \quad (9)$$

where $E_c|p^v = \{e \in E_c | hit(e,p) = v\}$.

The entropy (denoted *H*) represents a measure of the amount of uncertainty in the data and is usually computed as follows:

$$H(E_c) = -\sum_{i=1}^{n} p(t_i) \log p(t_i) \quad (10)$$

where *n* represents the number of latent types and *p(t)* represents the probability (relative frequency) of latent type *t* in E_c.

However in our case, an entity may have multiple types. Known as the multi-label learning problem, this poses difficulties for most learning and classification methods. The information gain - entropy based approach from the C4.5 decision tree algorithm is no exception [17]. To overcome this problem, we employ a modified version of the entropy proposed in [18] that considers multiple labels by introducing the probability of an entity not belonging to a certain type:

$$H(E_c) = -\sum_{i=1}^{n} ((p(t_i) \log p(t_i)) + (q(t_i) \log q(t_i))) \quad (11)$$

with *n* and *p(t)* as before and $q(t_i) = 1 - p(t_i)$ the probability of not having type t_i.

4 Evaluation

The approach we present in this paper has two major objectives: To provide transparency regarding the quality of the data accessed through the property-based paradigm and to improve the quality of the selected data by iteratively, and with user feedback, extending the property-based type definition with chosen properties. To evaluate how well these objectives have been fulfilled we performed the following experiment: Starting from different concepts presented in structured form with schemata on schema.org, as in the use case presented in Section 2, we build an initial type definition for each concept. This initial definition embodies typical properties most application developers require in order to program with each concept. It comprises the first four properties that have been most frequently annotated in ClueWeb12 for the corresponding schema.org schemata. The property-based data access is applied to these four properties and a set of entities from the BTC data corpus is selected. The quality score, precision and recall are computed for the selected entities. If the quality score is lower than 0.65 (our experiments have shown that a threshold of 0.65 brings satisfying data quality), a property is chosen based on its information gain. The user is asked whether this property belongs to the concept or not. We simulate the user feedback by relying on information from schema.org: If the property with the highest information gain is part of the schema that describes the corresponding concept on schema.org (considering synonymy), then the user feedback is positive. The type definition for the concept is in this case extended with this property and all entities having this property are kept. If, however, the property with the highest information gain is not

part of the schema, then it is considered a negative property and all entities not having this property are kept. The process is repeated until the quality score reaches the quality threshold. Using schema.org to simulate user feedback is convenient but it has some drawbacks that will be addressed in future work: Some properties that are part of schema.org may be irrelevant from a human perspective. At the same time, schema.org doesn't claim full completeness. In consequence one can't be sure that properties not being part of schema.org are negative properties.

In order to measure precision and recall, a gold standard is required. The gold standard represents, in this case, clear type information w.r.t. the concepts: In the context of movies, is a given entity a movie or not? We build the gold standard by bootstrapping on a set of 1000 seed entities that we know are of the concept type: We extract all rdf:type types for each of the seed entities. On average, about 500 types are found. Types that are not related to the concept or that are too general (e.g. owl:Thing or schema.org/CreativeWork) are manually pruned. In a second iteration, all entities having those types are selected and 100 entities are randomly chosen. Only those entities that, on manual inspection show the correct type are kept. Their rdf:type types are extracted, and unrelated or general types are again manually pruned. The process is repeated one more time. The resulting list of rdf:type values represents the description of a concept type according to the rdf:type property. Any entity that has one of the types in the list is considered to be of the respective type. Of course, only a subset of the actual expressions of a certain type is found. As a result, the precision and recall values computed on this gold standard underestimate the actual values.

Our system chooses key properties to improve the type definition based on information gain. As a baseline, we built Rand, a system choosing properties at random (without replacement). The randomization process is repeated 10 times for each property selection step. Average quality, precision and recall values are considered for each iteration. The property that is closest to the average scores of all 10 random picks is chosen to extend the definition for the next iteration.

We evaluated ProSWIP on multiple concepts from various fields, with different characteristics. For brevity reasons, in Table 5 we present the results on the example of three chosen concepts. The base iteration (0) is common to both systems and corresponds to the most frequent four properties used for annotating the corresponding schema in ClueWeb. For Movie, this iteration already produces good precision but it is quite restrictive in terms of recall. ProSWIP requires in this case 4 iterations to reach quality above 0.65 and perfect precision. With 0.93, precision is already very good after the first iteration. Further iterations isolate well defined movies from the ones with missing values. This in turn affects recall. Benefitting from high quality entity selection from the base iteration (78% of entities selected from the start are movies), the random approach is also able to obtain good results. Primarily guided by average scores and with high quality semantic feedback, the baseline method achieves 0.95 precision and a quality score of 0.59 after 4 iterations. Recall however is severely affected by the random choice of properties. For Music the base iteration is, with a precision of 0.44, of lower quality. Various types of entities are selected. The probability for the random selector to choose some irrelevant property is higher in this case. This is also reflected in the poor performance of Rand for Music. In contrast,

ProSWIP achieves the desired level of quality after only two iterations. For Books, the base also has low precision with negative consequences on the performance of Rand. The quality metric we introduced is highly correlated to precision on all experiments (Pearson's linear correlation coefficient of 0.94) denoting its expressiveness for the quality of the data selection. Precision rapidly increases towards values above 90%, showing the success of the whole approach.

Table 5. Quality, Precision and Recall for three chosen concepts and multiple iterations

Iteration	Quality(Q)		Precision		Recall	
Movies	ProSWIP	Rand	ProSWIP	Rand	ProSWIP	Rand
0	0.49	0.49	0.78	0.78	0.26	0.26
1	0.57	0.5	0.93	0.78	0.25	0.26
2	0.55	0.51	0.91	0.74	0.12	0.03
3	0.58	0.53	0.96	0.89	0.11	0.03
4	0.65	0.59	1	0.95	0.07	0
Music						
0	0.34	0.34	0.44	0.44	0.82	0.82
1	0.58	0.34	0.99	0.43	0.82	0.78
2	0.67	0.34	0.99	0.43	0.62	0.78
Books						
0	0.21	0.21	0.37	0.37	0.71	0.71
1	0.32	0.21	0.83	0.38	0.07	0.07
2	0.52	0.22	0.93	0.39	0.07	0.07
3	0.59	0.25	0.89	0.43	0.04	0.07
4	0.65	0.25	1	0.43	0.03	0.07

From a technical perspective, ProSWIP is a component implemented in Scala (www.scala-lang.org/), which maps variable names to properties from the BTC data set. While classical relational databases are not suitable for querying on RDF data, graph databases like Neo4j (www.neo4j.org/) have limited performance for our approach. In comparison, Lucene (lucene.apache.org/) has proven much faster in both the time needed for initially loading the data (building the index) as well as in terms of querying. With an off-the-shelf commodity computer with Intel I5-3550 quad-core CPU with 3.3 GHz. 32 GB RAM and 8.5 ms access hard drive, the index creation for the complete BTC data set took about 39 hours (only one core was used). The resulting index was about 1T in size including data. One simple entity search takes about 16 seconds. But the complete process of property-based data access may take up to hours as multiple queries, entity and property retrievals are being performed. It was possible to speed up the process by introducing caching mechanisms, for instance for the property synonymy dictionaries. Computing the quality, principal components, latent types and information gain for all properties on large data samples takes under 2 seconds. Nonetheless, we believe that in order to realize all operations in real-time a Lucene-based distributed index leveraging Hadoop is necessary.

5 Related Work

Property-based data access and its suitability for programming the Semantic Web has recently been discussed in [7, 19]. Challenges and open questions concerning a property based approach are discussed in these papers. Sharing their view, we inspect the

practical feasibility of such an approach and address one of the main challenges: The data quality problem.

Structural typing approaches are already employed in programming: Property-based interfaces have been studied for OO languages [20] or extensible record systems for different language settings [21, 22]. But additional challenges like discovering, comprehending and extending property sets to match the intended use arise in the context of linked open data.

From a broader perspective, systems like Tipalo [23] performing automatic typing for DBpedia entities are also relevant to our approach. Tipalo extracts types for entities based on their corresponding Wikipedia pages. But there are several entities in the LOD cloud having no article on Wikipedia that would hence remain untyped (there are about 14,199 diseases (International Statistical Classification of Diseases: http://www.who.int/classifications/icd/en/) most of them documented through PubMed but only about 3,000 of them featuring an actual article on Wikipedia). High precision knowledge bases like YAGO [24] relying on the Wikipedia category system and Infoboxes suffer from the same problem. In contrast, we build on structural similarity independent of all-encompassing information sources to find latent, contextually relevant types.

6 Conclusions and Outlook

We believe that property-based data access represents a cornerstone in programming with data from the Web. Our experiments show that such an approach suffers from quality problems that the end user is not even aware of. With an entity homogeneity-based quality metric and iterative feedback from the user on chosen properties, the level of quality for the selected data can be controlled. Being highly correlated to precision, the quality measure we introduced provides for transparency. With additional feedback on chosen properties, precision easily reaches values above 0.9, confirming the success of this approach.

The sparse nature of data in the LOD cloud severely affects recall. Leveraging high quality property-based definitions, the recall problem can be tackled: We plan to use properties that have been found suitable to extend the concept definition, not as filters, but as features for entity ranking on structural similarity. This should increase the robustness against missing values and have a positive effect on recall.

References

1. Barry, D., Stanienda, T.: Solving the Java Object Storage Problem. Computer 31, 33–40 (1998)
2. Carroll, J.J., Reynolds, D., Dickinson, I., Seaborne, A., Dollin, C., Wilkinson, K.: Jena: Implementing the Semantic Web Recommendations. In: Proc. WWW, New York, USA, pp. 74–83 (2004)
3. Völkel, M.: RDFReactor – From Ontologies to Programmatic Data Access. In: Proc. ISWC (2005)

4. Tratt, L.: Compile-time meta-programming in a dynamically typed OO language. In: Proc of the Dynamic Languages Symposium (DLS), San Diego, California, USA (2005)
5. Stock, W.G.: Concepts and Semantic Relations in Information Science. Journal of the American Society for Information Science and Technology 61, 1951–1969 (2010)
6. Scheglmann, S., Gröner, G.: Property-based Typing for RDF-Access. In: Proc. of Workshop on Programming the Semantic Web, Rio de Janeiro, Brasil, pp. 4–7 (2012)
7. Scheglmann, S., Groener, G., Staab, S., Lämmel, R.: Incompleteness-aware programming with RDF data. In: Proc. of Workshop on Data Driven Functional Programming (DDFP), vol. 11 (2013)
8. Cardelli, L.: Structural subtyping and the notion of power type. In: Proceedings of ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages POPL 1988, pp. 70–79. ACM Press (1988)
9. Cafarella, M.J., Etzioni, O.: Navigating Extracted Data with Schema Discovery. In: Proc. of Int. Workshop on Web and Databases (WebDB), Beijing, China (2007)
10. Jaccard, P.: Nouvelles recherches sur la distribution orale. Bulletin Societe Vaudoise des Sciences Naturelles 44, 223–270 (1908)
11. Hancock, T., Jiang, T., Li, M., Tromp, J.: Lower Bounds on Learning Decision Lists and Trees. Information and Computation 126, 114–122 (1996)
12. Quinlan, J.R.: C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers Inc., San Francisco (1993)
13. Quinlan, J.R.: Simplifying decision trees. Int. Journal of ManMachine Studies 27, 221–234 (1987)
14. Gottron, T., Knauf, M., Scheglmann, S., Scherp, A.: A Systematic Investigation of Explicit and Implicit Schema Information on the Linked Open Data Cloud. In: Cimiano, P., Corcho, O., Presutti, V., Hollink, L., Rudolph, S. (eds.) ESWC 2013. LNCS, vol. 7882, pp. 228–242. Springer, Heidelberg (2013)
15. Jolliffe, I.: Principal Component Analysis, 2nd edn. Springer Series in Statistics (2002)
16. Ball, G., Hall, D.: ISODATA: A novel method of data analysis and pattern classification (1965)
17. Tsoumakas, G., Katakis, I.: Multi Label Classification: An Overview. Int. Journal of Data Warehousing and Mining 3, 1–13 (2007)
18. Clare, A., King, R.D.: Knowledge Discovery in Multi-label Phenotype Data. In: Siebes, A., De Raedt, L. (eds.) PKDD 2001. LNCS (LNAI), vol. 2168, pp. 42–53. Springer, Heidelberg (2001)
19. Scheglmann, S., Scherp, A., Staab, S.: Declarative Representation of Programming Access to Ontologies. In: Simperl, E., Cimiano, P., Polleres, A., Corcho, O., Presutti, V. (eds.) ESWC 2012. LNCS, vol. 7295, pp. 659–673. Springer, Heidelberg (2012)
20. Gil, J.Y.: Whiteoak: Introducing Structural Typing into Java, pp. 73–89 (October 2008)
21. Bracha, G., Lindstrom, G.: Modularity meets inheritance. In: Proc. of Int. Conf. on Computer Languages, pp. 282–290. IEEE (1992)
22. Kiselyov, O., Lämmel, R., Schupke, K.: Strongly typed heterogeneous collections. In: Proc. of the SIGPLAN Workshop on Haskell, pp. 96–107 (2004)
23. Gangemi, A., Nuzzolese, A.G., Presutti, V., Draicchio, F., Musetti, A., Ciancarini, P.: Automatic Typing of DBpedia Entities. In: Cudré-Mauroux, P., Heflin, J., Sirin, E., Tudorache, T., Euzenat, J., Hauswirth, M., Parreira, J.X., Hendler, J., Schreiber, G., Bernstein, A., Blomqvist, E. (eds.) ISWC 2012, Part I. LNCS, vol. 7649, pp. 65–81. Springer, Heidelberg (2012)
24. Suchanek, F.M., Weikum, G.: YAGO : A Core of Semantic Knowledge Unifying WordNet and Wikipedia. In: Proc. of WWW, Banff, Canada (2007)

Simplified OWL Ontology Editing for the Web: Is WebProtégé Enough?

Matthew Horridge, Tania Tudorache, Jennifer Vendetti, Csongor I. Nyulas, Mark A. Musen, and Natalya F. Noy

Stanford Center for Biomedical Informatics Research
Stanford University, Stanford, CA, 94305, USA
{horridge,tudorache,vendetti,nyulas,musen,noy}@stanford.edu

Abstract. Ontology engineering is a task that is notorious for its difficulty. As the group that developed Protégé, the most widely used ontology editor, we are keenly aware of how difficult the users perceive this task to be. In this paper, we present the new version of WebProtégé that we designed with two main goals in mind: (1) create a tool that will be easy to use while still accounting for commonly used OWL constructs; (2) support collaboration and social interaction around distributed ontology editing as part of the core tool design. We designed this new version of the WebProtégé user interface empirically, by analysing the use of OWL constructs in a large corpus of publicly available ontologies. Since the beta release of this new WebProtégé interface in January 2013, our users from around the world have created and uploaded 519 ontologies on our server. In this paper, we describe the key features of the new tool and our empirical design approach. We evaluate language coverage in WebProtégé by assessing how well it covers the OWL constructs that are present in ontologies that users have uploaded to WebProtégé. We evaluate the usability of WebProtégé through a usability survey. Our analysis validates our empirical design, suggests additional language constructors to explore, and demonstrates that an easy-to-use web-based tool that covers most of the frequently used OWL constructs is sufficient for many users to start editing their ontologies.

1 Introduction

"Protégé is too difficult to use!" The Protégé team hears this sentiment from our users all too often. As we observe many of them grapple with the difficulties of learning a highly expressive logic-based ontology language such as OWL, we see how onerous ontology development can be. Other studies on cognitive complexity of ontology development bear out these observations [1].

Developers of tools for ontology browsing and editing have faced the dilemma: On the one hand, we want to support international standards, such as OWL 2, fully in order to ensure interoperability [2]. On the other hand, we want to make sure that both beginners and experts alike can develop ontologies easily.

In this paper, we report on our design and evaluation of a major new release of WebProtégé, a web-based version of Protégé that uses the OWL API [3] and

H. Alani et al. (Eds.): ISWC 2013, Part I, LNCS 8218, pp. 200–215, 2013.

that provides support for editing OWL 2 ontologies. We released this new version of WebProtégé in January 2013, and we had two main goals for the design of this version: (1) create a tool that will be easy to use while still accounting for commonly used OWL constructs; (2) support distributed ontology editing, collaboration, and interaction as part of the core tool design. The new WebProtégé serves as a "Google docs" environment for ontologies, enabling users to upload their ontologies, to initiate new projects, and to invite their collaborators to participate in the development. We have created a cut-down user interface in WebProtégé, which makes creating new classes or updating information as simple as filling out a web form. This interface is the default interface that WebProtégé users see when they create or upload an OWL ontology. The users have the option of enabling more advanced features. In this paper, we discuss the design and evaluation of the choice of language constructs supported in this default interface, assuming that the default interface is what the majority of our users will see.

We address the following research questions in this paper: (1) Is there a subset of OWL that accounts for the majority of term descriptions used by ontology developers in various scientific domains? (2) How do we design a user interface that enables efficient editing of the most common constructs while providing an opportunity for the more expert users to access as much of the advanced features as possible?

In order to address these questions, we start by analysing a corpus of 330 publicly available ontologies in BioPortal [4,5] to determine which OWL constructs ontology developers use most frequently (Section 4). We use the results of this analysis to determine which set of features to include in the default configuration of WebProtégé. We evaluate this new release in two ways. First, we evaluate the *coverage* of the constructs supported by the user interface by analysing the aggregated information about the ontologies that WebProtégé users uploaded to the WebProtégé server. This new corpus constitutes a set of ontologies that were created elsewhere and thus presents a "naturally occurring" corpus of ontologies. Second, we conducted a survey of the users of the new tool in order to evaluate the usability of the tool; to understand what the users like and do not like about the tool; and to gauge whether or not the users feel limited by the default interface or whether they feel that they can perform all of the editing tasks that they need to perform (Section 6).

This paper makes the following contributions: (1) We present an empirical methodology for developing an easy-to-use ontology editor based on analysing a large training corpus of ontologies. (2) We evaluate the *language coverage* provided the user interface in WebProtégé by analysing the OWL constructs in a test corpus of 230 ontologies that WebProtégé users have created in other tools and uploaded to WebProtégé. (3) We evaluate the *usability* of WebProtégé through a usability questionnaire that close to 20% of WebProtégé users have answered.

Many of the lessons that we learn from designing and evaluating WebProtégé are not specific to our tool. Indeed, our paper analyses the broader question

of how we can use a principled approach to make ontology editing easier and whether simplified ontology editing is indeed possible, practical, and useful.

2 Preliminaries

In the work presented here, we deal with ontologies written in the Web Ontology Language (OWL), and more specifically OWL 2, its latest version [6]. Throughout the rest of this paper we refer to OWL 2 simply as OWL. In this section, we present the main OWL terminology that is useful in the context of this paper. We assume that the reader has basic familiarity with ontologies and OWL.

OWL and Ontologies. An OWL ontology is a set of *axioms*. Each axiom makes a *statement* about the domain of interest. The building blocks of axioms are *entities* and *class expressions*. Entities correspond to the important terms in the domain of interest and include *classes*, *properties*, *individuals*, and *datatypes*. Properties may be subdivided into *object*, *data*, and *annotation* properties. The *signature* of an ontology is the set of entities that appear in that ontology. OWL is a highly expressive language and features a rich set of class constructors that allow entities to be combined into more *complex class expressions*. As a convention, we use the letters A and B to stand for class names and the letters C and D to stand for (possibly complex) class expressions. In this paper, we largely focus on subclass axioms SubClassOf(C, D), equivalent class axioms EquivalentClasses(C, D), disjoint classes axioms DisjointClasses(C, D) and annotation assertions AnnotationAssertion(P, A, v). We refer to SubClassOf, EquivalentClasses and DisjointClasses axioms as *logical* axioms and AnnotationAssertion axioms as *non-logical* axioms. We also focus on two broad types of class expressions: (1) class expressions that we loosely term *existential restrictions*, which specify the existence of relationships between individuals and by which we mean ObjectSomeValuesFrom(R, C), DataSomeValuesFrom(R, C), ObjectHasValue(R, a), and DataHasValue(R, l) restrictions; and (2) class expressions that we term *universal restrictions*, by which we mean ObjectAllValuesFrom(R, C) and DataAllValuesFrom(R, C) restrictions.

Frame-Based Views of Ontologies. Even though an OWL ontology is simply a set of axioms, few ontology-development environments choose to display ontologies as lists of axioms. Most environments are *entity-centric* and revolve around the idea of editing *descriptions* of entities. In essence, when an entity is selected in a tool like Protégé, the tool presents (a partial view of) the subset of axioms that describe or define that entity. We call such a subset of axioms an *entity-frame*, or more specifically a *class-frame* for a class and so on. In this work we focus on class-frames, which we define as follows:

Definition 1 (Class-Frame). *For a class A in the signature of an ontology $\mathcal{O}$ the class-frame for A w.r.t. $\mathcal{O}$ is the subset-maximal set of axioms $\mathcal{S} \subseteq \mathcal{O}$ where each axiom in $\mathcal{S}$ is of the form SubClassOf(A, C), EquivalentClasses(A, C), DisjointClasses(A, C) and AnnotationAssertion(P, A v), where C is a (possibly complex) class expression, P is an annotation property, and v is an annotation value (literal, IRI or anonymous individual).*

OBO and OWL. In the world of biomedical ontologies, there is another, widely used language, called OBO [7]. There is a close relationship between OBO and OWL 2, and it is possible to translate faithfully the logical aspects of an OBO ontology into an OWL 2 ontology [8]. For the purposes of the work here we therefore view OBO as a syntactic variant of OWL 2.

3 An Overview of WebProtégé

This section presents a high level overview of WebProtégé and its salient features. The main purpose of this section is to provide a context for the discussion on our empirically driven user interface (UI) design in Section 4.

WebProtégé is a web-based, multi-user, collaborative editor for OWL ontologies. The main document unit in WebProtégé is a *project*, which is a set of ontologies plus metadata, sharing settings and UI settings. Users create their own projects, which are hosted on our servers at Stanford.[1] They either start by creating a new ontology, or they start by uploading a set of existing ontologies that they have already worked on. Having created a project, a user then "shares" this project, adding the names of her collaborators to the list of those who can edit her ontology. Now, any time the user or any of her collaborators logs into WebProtégé, she can see her ontology under development. As one of the users creates or edits the ontology, others can see the changes immediately. They can comment on the changes and carry out discussions in the tool—with the discussions linked to the class that they are discussing. If they log in after a few days, they can see the summary of changes to the ontologies and to the classes on their "watch list." As the project matures, they can invite others to participate and to comment, or choose to publish the ontology in a public repository for the broader community to use. They can download any revision of their ontology and process it using any other OWL tools such as reasoners, visualisation, and query tools.

Figure 1 shows a screenshot of the main editing interface in WebProtégé. The left pane consists of a tree for navigating the class hierarchy and for selecting a class frame for editing. The middle pane captures a subset of the selected class frame. We provide a precise description of and the rationale for what this frame captures in Section 4. The right pane in Figure 1 contains tools for collaboration. In particular, it shows a threaded list of issues and discussions and a live activity feed. Users can configure all elements in the interface, augment it with different views or reconfigure it completely to suit their needs.

The centre pane in Figure 1 is the main editing form for class frames. The form is composed of fields which constitute tables of property–value pairs. The fields feature auto-completion for property, class, individuals and datatype names. The auto-completion is type sensitive: It will offer only the types of entities that can be entered based on the information in the ontology up to this point. For example, the auto-completion prevents the user from entering datatypes as fillers for object property restrictions. In terms of OWL, one row in the table corresponds

[1] Users can also set up local WebProtégé installations if they have a desire to do so.

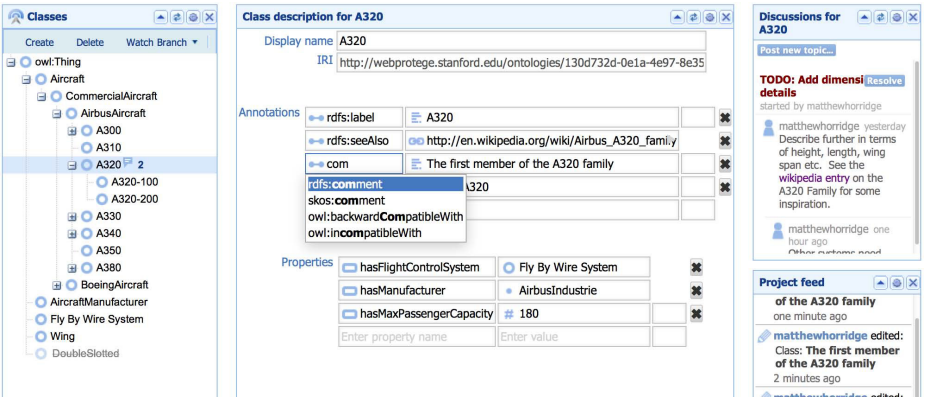


Fig. 1. The main editing interface in WebProtégé. The lefthand pane presents the class tree, indicating which classes have discussions attached to them. The middle pane presents the class frame. The righthand pane shows the discussions for the class and the live feed of changes.

to one or more axioms. In the example in Figure 2, the row hasFlightControlSystem and FlyByWireSystem corresponds to the axiom SubClassOf(:A320, ObjectSomeValuesFrom(:hasFlightControlSystem :FlyByWireSystem)).

A key feature of the WebProtégé UI is that it minimises the distinctions that users have to make explicitly. For example, in previous versions of the tool [9], when a user created a new property, she had to decide explicitly whether the property was an object or a data one. Similarly, when entering class expressions the user had to make various choices such as choosing between SomeValuesFrom and AllValuesFrom restrictions, and between SomeValuesFrom and HasValue restrictions. In the WebProtégé UI, we use simple and reliable heuristics to determine the type of property and the type of restriction that the user creates based on the fillers that she specifies. Figure 2 displays a class description that has mixed use of data and object properties. It also contains mixed use of different types of class expressions, individuals, and data values: the first row corresponds to an ObjectSomeValuesFrom class expression whose filler is a class, the second row an ObjectHasValue class expression whose filler is an individual, and the third row a DataHasValue class expression whose value is an integer literal. At no point when entering the information shown in Figure 2 has the user *explicitly* had to decide upon and choose the types of class expressions, or decide upon and choose the types of properties—the system determines these distinctions in a straightforward but highly effective way. Finally, this UI also supports a kind of on-the-fly object creation and type inference. In the fourth row in Figure 2 the user wants to specify a new type of flap for the class (aircraft) that she is describing. However, hasFlap is a new property name. In this case, the system accepts the new property name, warns the user that it is new (in case the user has simply made a typo) and allows her to move on to specify a filler. In this case, she specifies a new class (DoubleSlottedFlap). Once the user enters this information, WebProtégé

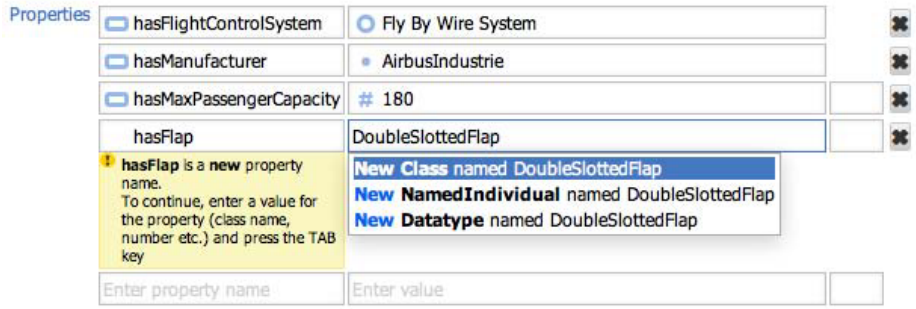


Fig. 2. Property–value pairs being edited. The class frame in the figure contains mixed object and data property usage. It also contains a mix of ObjectSomeValuesFrom, ObjectHasValue and DataHasValue class expressions. The auto-completion box prompts the user to create new entities where necessary. We eliminated the need to choose explicitly between object and data properties; we determine property types based on filler values.

creates the necessary declarations of the appropriate type and generates the class expressions and axioms under the hood.

In addition to editing logical information, WebProtégé provides support for describing extra-logical information about entities through OWL annotations. These annotations are part of the class frame (Figures 1 and 2). WebProtégé provides auto-completion support that allows users to reuse annotation vocabulary from well known metadata sets such as DublinCore and SKOS (Figure 1).

4 The WebProtégé Profile (WPP)

The following are our main high level design goals for WebProtégé: (1) to provide an ontology-editing infrastructure with zero installation and zero setup costs, (2) to provide a framework that supports multiple editors, commenters and viewers to work simultaneously on the same ontology; (3) to provide a simple UI that allows novices and experts alike to enter information in a way that is comfortable for them. Developing WebProtégé as a Web-app achieves the first goal and goes some of the way in supporting collaboration. In this section, we look at the simple UI that WebProtégé provides, what exactly it can represent and, how we arrived at this current design. We call the set of language features supported by the UI the *WebProtégé Profile* (WPP).

Definition of the WebProtégé Profile

Although WebProtégé supports the editing of class, property, and individual frames, we focus our discussion on class frames. We focus on class frames because property frames are somewhat simpler than class frames, with fewer design choices to make, and individual frames are themselves similar to class frames.

The default class frame editor in WebProtégé supports editing class frames defined as follows.

Definition 2 (WPP). *A WebProtégé Profile class frame for a class A in the signature of an ontology $\mathcal{O}$ is the subset-maximal set of axioms $\mathcal{S} \subseteq \mathcal{O}$ such that each axiom in $\mathcal{S}$ conforms to the following grammar, where non-terminals are shown in bold, terminals are shown in a regular font-weight surrounded by single quotes, choices are indicated with a bar, zero or more items are shown in curly brackets. The non-terminals* **Class**, **ObjectProperty**, **DataProperty**, **AnnotationProperty**, **NamedIndividual**, **Datatype**, **Literal** *and* **IRI**, *are defined as they appear in the OWL 2 Structural Specification.*

ClassFrame := {**ClassFrameAxiom**}
ClassFrameAxiom := *'SubClassOf'* *'('* *A* **ClassExpression** *')'* |
'AnnotationAssertion' *'('* **AnnotationProperty** *A* **AnnoValue** *')'*
ClassExpression := **Class** |
'ObjectIntersectionOf' *'('* **ClassExpression ClassExpression** {**ClassExpression**} *')'* |
'ObjectSomeValuesFrom' *'('* **ObjectProperty**, **Class** *')'* |
'ObjectSomeValuesFrom' *'('* **ObjectProperty**, *'{'* **NamedIndividual** *'}'* *')'* |
'ObjectHasValue' *'('* **ObjectProperty**, **NamedIndividual** *')'* |
'DataSomeValuesFrom' *'('* **DataProperty**, **Datatype** *')'* |
'DataSomeValuesFrom' *'('* **DataProperty**, *'{'* **Literal** *'}'* *')'* |
'DataHasValue' *'('* **DataProperty**, **Literal** *')'* |
'ObjectMinCardinality' *'('* *'1'* **ObjectProperty**, **Class** *')'* |
'DataMinCardinality' *'('* *'1'* **DataProperty**, **Class** *')'*
AnnoValue := **Literal** | **IRI**

Definition 2 (WPP) precisely represents the language that is supported by the default class frame editor in WebProtégé. We chose what to include in the Definition 2 (WPP) based on (1) an empirical analysis of commonly used axiom types and class constructors in a large ontology corpus, and (2) commonly reported errors [10,11] that are made by novices when building OWL ontologies. The corpus analysis provided information on which constructs we should support. The error analysis helped us to decide which decisions we should take out of the hands of novice users.

An Analysis of Constructs from the BioPortal Ontology Corpus

BioPortal is a community-based repository of biomedical ontologies [4].[2] At the time of writing, it contains more than 330 public ontologies with almost six million terms in them. We used OWL and OBO ontologies from BioPortal as our corpus to analyse the commonly used OWL constructs.

While BioPortal contains only the ontologies that are developed by researchers and practitioners in biomedicine, it is still an attractive corpus for a general-purpose analysis for the following reasons: First, ontologies in BioPortal vary

[2] `http://bioportal.bioontology.org`

Table 1. Class frame axioms and class constructor occurrences in the BioPortal corpus. The corpus contains 261 ontologies in OWL and OBO. Pn represents the n^{th} percentile number of occurrences of a particular construct. For example, a P25 of 185 for SubClassOf axioms means that 25% of ontologies contain 185 SubClassOf axioms or less. The category Existential includes ObjectSomeValuesFrom, DataSomeValuesFrom, ObjectHasValue, and DataHasValue class expressions. The category Universal includes ObjectAllValuesFrom and DataAllValuesFrom class expressions. MinCardinality, MaxCardinality and ExactCardinality combine both object and data cardinality restrictions.

Constructor Type	# of ontologies	% of ontologies	# occurrences of constructors				
			P25	P50	P75	P90	Max #
SubClassOf	243	93.1	185	521	2705	12,309	847,755
EquivalentTo	80	30.7	4	16	61	403	73,461
DisjointWith	82	31.4	3	28	158	673	56,192
Existential	162	62.1	37	157	1,461	9,651	641,123
Universal	45	17.2	4	22	49	145	22,371
Object Union	64	24.5	3	7	20	65	387
Object Complement Of	19	7.3	1	4	15	35	99
Object One Of	8	3.1	1	1	4	4	5
MinCardinality	28	10.7	1	3	5	14	1,305
MaxCardinality	10	3.8	1	3	10	110	967
ExactCardinality	23	8.8	4	10	20	23	257

greatly in size and expressivity [12]. Second, these ontologies are naturally occurring ontologies, and they are developed by a wide range of groups and ontology engineers. Finally, biomedical ontologies account for a large fraction of ontologies under development in tools such as Protégé. Therefore, it seems reasonable that, if we can provide a UI that accommodates a large proportion of the BioPortal ontologies, then that UI will also satisfy a large number of potential WebProtégé users.

Materials and Method. We accessed BioPortal on August 31, 2012 using the NCBO Web services API [5]. We downloaded all OWL compatible (OWL plus OBO) ontologies. There were 261 such ontologies. We used the OWL API (version 3.4.0) to parse and analyse each ontology. We recorded the number and kinds of class frame axioms (SubClassOf, EquivalentClasses, DisjointWith) for each ontology, as well as the number of occurrences of the different kinds of OWL class expressions.

Results. Table 1 shows the occurrences of class frame axioms and class expressions. For each type of constructor, the table presents the number and percentage of ontologies that contain that constructor and the 25th, 50th, 75th, 90th percentile values (over the ontologies containing that constructor), and maximum occurrences per ontology.

Analysis. It is clear from Table 1 that SubClassOf is the dominant form of axiom type. Most ontologies (93%) contain these types of axioms. By contrast, DisjointClasses and EquivalentClasses axioms are present in just under one third of the ontologies in the corpus. Moreover, SubClassOf axioms are present in large

numbers when compared to EquivalentClasses axioms and DisjointClasses axioms—on average two orders of magnitude more. The picture for class constructors is similar: The dominant form of class constructor is Existential restriction (including ObjectSomeValuesFrom, DataSomeValuesFrom, ObjectHasValue, and DataHasValue). Nearly two thirds of ontologies contain axioms which use one or more type of Existential restriction. By contrast, Universal restrictions are used in 17% of ontologies and many of the other class constructors in fewer than 10% of ontologies. Furthermore, on average the occurrences of Existential restrictions are two orders of magnitude greater than the occurrences of Universal restrictions which are themselves on average two orders of magnitude greater than occurrences of all other types of class constructors. Finally, we observed that some ontologies contain MinCardinality 1 restrictions as a syntactic variant of Existential restrictions.

The stand-out axioms and class constructors from the BioPortal corpus are SubClassOf axioms and Existential restriction class expressions. We therefore decided to focus on these constructs in the simplified WebProtégé UI.

5 Evaluating Coverage

One of the features in the new WebProtégé is the ability of users to upload their ontologies to the WebProtégé server. Users created these ontologies with other tools or download them from the Web. Since we released WebProtégé, users have uploaded 230 ontologies to our server.[3] This corpus represents the naturally occurring ontologies that WebProtégé users want to work with. Therefore, this collection of ontologies offers a rich source of data that we can use to empirically drive forward the development of the tool. In this section, we analyse this corpus to assess how well the simple profile defined in Definition 2 covers the ontologies that people actually want to edit in WebProtégé. We then discuss how we can use this information to evolve WebProtégé in the future.

For the purposes of this evaluation we also examine two extensions of WPP. The first, WPP-Dis, extends WPP with DisjointClasses axioms and is defined in Definition 3, while the second, WPP-DisEq, extends WPP with DisjointClasses and EquivalentClasses axioms and is defined in Definition 4. The motivation for these extensions is to determine how many class frames are excluded from being represented in the simplified WebProtégé UI because of the fact that it does not display DisjointClasses axioms or EquivalentClasses axioms.

Definition 3 (WPP-Dis). *A WebProtégé Profile class-frame with DisjointClasses axioms (*WPP-Dis*) for a class A is defined as in Definition 2 but with the grammar augmented with the following production rule below, where the non-terminals* ***ClassFrameAxiom*** *and* ***ClassExpression*** *are specified in Definition 2.*

ClassFrameAxiom := *'DisjointClasses' '(' A* ***ClassExpression*** *')'*

[3] The WebProtégé privacy policy prevents us from making this corpus available in its raw form. Moreover, the analysis that we conducted looks at aggregated data and ontology constructs from a structural point. We do not critically examine any domain content of any projects.

Definition 4 (WPP-DisEq). *A WebProtégé Profile class-frame with* ***DisjointClasses*** *axioms and* ***EquivalentClasses*** *axioms (*WPP-DisEq*) for a class A is defined as in Definition 2 but with the grammar augmented with the following production rule, where the non-terminals* ***ClassFrameAxiom*** *and* ***ClassExpression*** *are specified in Definition 2.*

ClassFrameAxiom := *'DisjointClasses' '(' A* ***ClassExpression*** *')'* |
'EquivalentClasses' '(' A ***ClassExpression*** *')'*

Materials and Method. On May 6th 2013 the version of WebProtégé hosted at Stanford contained 519[4] non-empty projects. Of these, 230 projects were created by users who uploaded their existing non-empty ontologies (the remaining projects were edited from scratch in WebProtégé). We parsed each non-empty ontology in the set of 230 using the OWL API version 3.4.3. We examined the classes in the signature of each ontology according to Definition 1 to determine which of them had WPP class frames satisfying Definition 2 (i.e. which of them can be represented by the simple UI). For each ontology, we measured the coverage in terms of the percentage of WPP, WPP-Dis and WPP-DisEq class frames.

Results. Figure 3 shows a plot of class frames over the WebProtégé ontology corpus. Each bar represents one ontology (one project) with a non-empty class signature. The full length of a bar indicates the number of general class frames (Definition 1) in the ontology represented by that bar. Each bar is divided into four segments (note that zero size segments are not visible in the plot) representing the WPP class frames that satisfy Definition 2 (painted white); the WPP-Dis class frames that satisfy Definition 3 (painted grey); the WPP-DisEq class frames that satisfy Definition 4 (painted with a hatch effect); and the class frames that are neither WPP, WPP-Dis or WPP-DisEq frames (painted black). Figure 4 shows a plot representing the class frame coverage over the complete set of ontologies in the WebProtégé corpus. Each line represents class frames falling into the WPP (solid black), WPP-Dis (dashed black) and WPP-DisEq (dashed grey) profiles, with the plot showing the relationship between the number of ontologies and percentage of frames covered.

Analysis. Broadly speaking, the simple UI in WebProtégé can represent the majority of class frames in the majority of ontologies in this corpus—there are 108 ontologies (or 47% of the corpus) for which it can present 100% of the class frames, and a further 12 ontologies (coming to just under 60% combined) for which it can present 90% of the class frames. Figure 4 plots the coverage of ontologies by the WPP as a black solid line. Each point on the line represents an ontology and the percentage of its terms covered by the profile. Combined, there are 156 ontologies (just under 70% of the corpus) for which the WPP can

[4] This number does not include a handful of projects created by the authors and colleagues at Stanford that we excluded from this analysis so as not to bias results. We also excluded several copies of the "pizza" ontology, which is a tutorial ontology containing most OWL 2 constructs.

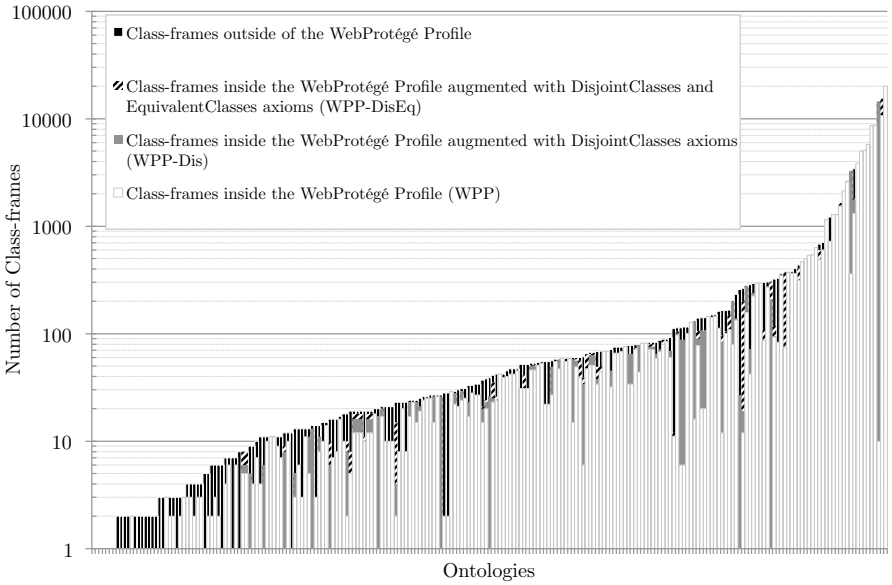


Fig. 3. WPP, WPP-Dis and WPP-DisEq class frames by ontology. Each bar represents one ontology with a signature size greater than zero. The total height of any given bar represents the total number of classes in the signature of the ontology.

capture 75% or more of the class frames in each ontology. These results are acceptable for a number of reasons: (1) the simple UI *completely* caters for a large fraction (roughly half) of the users that decided to edit their ontologies in WebProtégé—we expect a mix of novices and experts to use our tool and therefore the simple UI need not cater for everybody; (2) we do not expect the UI to cover the *whole* corpus—if it did, it would ultimately have to capture the full expressivity of OWL; and (3) it is conceivable that in collaborative settings there will be a cross-section of users, with less experienced users working with more experienced users. In these case, less experienced users may well prefer to edit the majority of the class frames in the ontology in the simple UI, while the more experienced users take care of class definitions that cannot be expressed in this UI.

At the lower end of the scale, there are three ontologies (1% of the corpus) for which the WPP *cannot represent any* class frames at all, and 38 ontologies (16% of the corpus) for which it can only represent 50% of class frames or less on average. A closure examination reveals that all of the ontologies that do not contain *any* class frames that are captured by the WPP, or ontologies that contain very low numbers of captured class frames, are like this because they contain DisjointClasses axioms. Looking at Figure 3, there are several long grey bands. These bands represent ontologies with large numbers class frames that are *not* captured by WPP (Definition 2) but *are* captured by WPP-Dis (Definition 3).

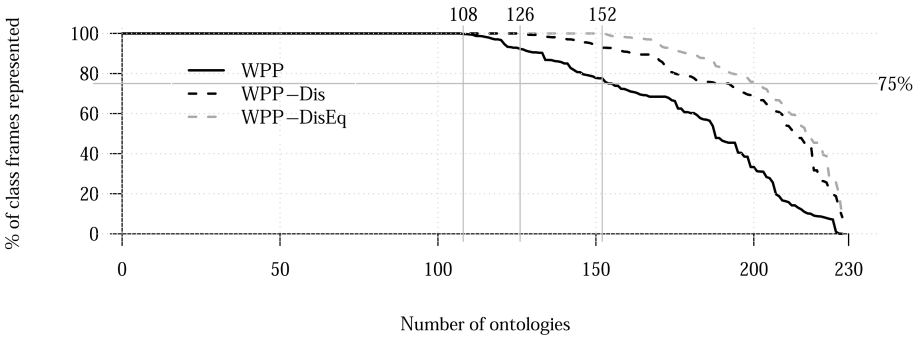


Fig. 4. Coverage of ontologies in the WebProtégé corpus. Each point on the X axis corresponds to an ontology; the Y axis plots the percentage of class frames that are covered in each profile. Each line represents a profile. The area under a line represents the number of class frames covered over the whole corpus by the profile represented by that line. As we extend the WPP profile to the WPP-DIS profile and then to the WPP-DISEQ profile, the number of ontologies with 100% coverage increases.

In other words, they represent class frames which utilise DisjointClasses axioms. The effect of admitting DisjointClasses axioms to WPP is highlighted in the difference between plots in Figure 4. The plot shifts to the right, representing the ontologies for which there is 100% coverage, increasing by almost 20 ontologies with the addition of disjoint axioms. Put simply, admitting DisjointClasses axioms by supporting them in the simple UI would allow many more ontologies to be fully captured. In a similar vein, looking at the difference between the dashed black and dashed grey plots in Figure 4, it is clear that either including or excluding EquivalentClasses axioms has a noticeable effect on the number of ontologies all of whose class frames can be captured by a UI supporting these type of axioms—the number jumps from 126 (55%) to 152 (67%).

6 Evaluating Usability

In order to evaluate the usability of WebProtégé and to understand what the users like and dislike about the interface, we have conducted a survey among those users who had a chance to try the new WebProtégé design, either in its beta phase or after the official release.

Materials and Method. We have designed the survey using SurveyMonkey®. The survey contained three types of questions: (1) qualification: the survey rules and the question asking respondents to confirm that they have had a chance to use the new version of WebProtégé. (2) usability: the questions about the user experience with the tool. (3) demographics: the questions about the user level of expertise and the type of projects that they were working on.

The survey included six usability questions [13] on a 5-point Likert scale (Figure 5) as well as free-text questions for feedback about the tool. Specifically, we

asked what users liked about WebProtégé, what they felt needed to be improved, and what type of content they wanted to enter but could not. The last question in particular was designed to gain insight on what important OWL 2 constructs we failed to include in the WebProtégé Profile (WPP—Definition 2).

We emailed the survey link to all the users with the account on the WebProtégé server, to the Protégé support mailing lists, and posted the link on the Protégé social media channel. The survey was open for seven days. While the survey was completely anonymous, participants had the option of entering into a draw for a $25 gift card as a reward for their participation. Contact details for this were collected via a completely separate Web-form to preserve anonymity.

Results. We received 55 responses from those who confirmed that they have used the new version of WebProtégé; 23 of the respondents chose to enter the draw for the gift card. Given that the WebProtégé change history lists contain changes or actions from distinct 288 users, our survey contains responses from 19% of the users who tried or used the system. The vast majority of respondents (90%) followed the link to the survey from the direct invitation email. Others followed the link in one of the Protégé mailing-list posts (6%), with the remainder using the links on Web sites. Among the respondents, 70% were from academia and 17% from industry, with the remainder from government, museums, and other organisations. We received responses from across the world, with the largest share of contributions from Europe (40%) and North America (40%).

As far as users' self-reported level of expertise with ontologies and OWL is concerned, on a 5-point Likert scale (1-Beginner and 5-Expert), the average expertise in ontologies was 2.96 (1:15%, 2:21%, 3:27%, 4:27% and 5:10%) and the average expertise in OWL was 2.7 (1:21%, 2:19%, 3:35%, 4:21% and 5:4%). All respondents have performed some content editing in WebProtégé, either editing an ontology (64%), uploading an ontology (57%), downloading an ontology (45%), defining sharing settings (38%), and other actions. 17% of the respondents participated in collaborative editing.

Figure 5 shows the distribution of answers to the usability questions on a 5-point Likert scale (1-Strongly disagree to 5-Strongly agree). Overall, 78% of the users agreed that they were satisfied with WebProtégé; 75% agreed that it was easy to use and 70% agreed that it was easy to learn. We have also looked separately at the results from the self-identified experts in ontology development and self-identified beginners. On all questions in Figure 5, experts were slightly more positive than the overall cohort. Beginners found collaborative features less useful than the overall group. In general, as can be seen from Figure 5 the responses for all questions are skewed towards "agree" and "strongly agree".

We asked the survey respondents to identify specific content that they wanted to enter but were not able to enter in the simplified interface. Of the 55 respondents, one missed the ability to directly "create anonymous classes" and one wanted to "create logical expressions". The other 53 respondents did not indicate any specific constructs that they were not able to enter in the simplified UI.

Analysis. While our survey results are limited to the early adopters of the tool the results are encouraging. The overall skew of the usability question responses

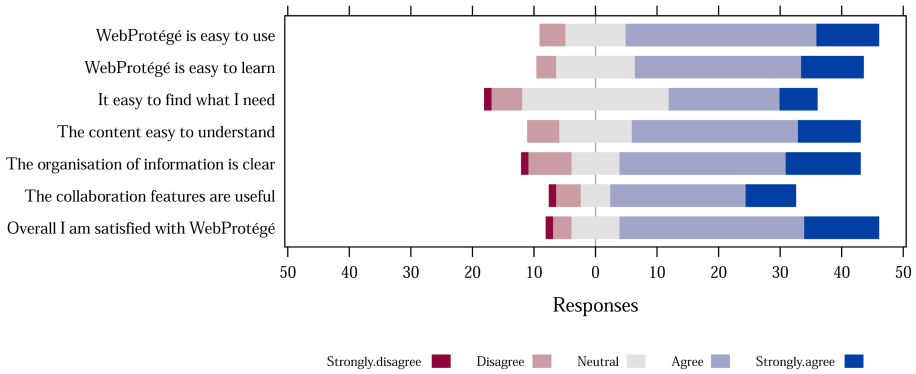


Fig. 5. A plot of responses to questions in the usability questionnaire. Blocks on the left of the centre-line (dark/light red) represent negative responses (strongly disagree/disagree). Blocks on the right of the centre-line (mid/dark blue) represent positive responses (agree/strongly agree). Blocks on the centre-line represent neutral responses. The size of each block is proportional to the number of responses.

towards "agree" and "strongly agree" indicates that users feel comfortable using the tool. The fact that only two users commented that they could not enter complex class expressions seems to indicate that users do not necessarily feel limited by the simple UI. Finally, given the mix of respondent expertise in ontologies and OWL, we believe that the simple UI might be capturing the best of both worlds: simple enough for novices to learn and to use, yet powerful enough for experts to do their job. Indeed, when using the interface ourselves to develop ontologies, we observed that we ourselves appreciated the many shortcuts that WebProtégé now provides as they made our work more efficient.

7 Discussion

In the past decade, researchers developed a number of Web-based ontology-development tools, such as OntoWiki [14], MoKi [15], Neologism [16], PoolParty [17], TopBraid EVN and others. Similarly, semantic wikis add semantic capabilities to traditional wikis. These semantic wikis [18] usually associate a Web page with a particular instance in the ontology, and the semantic Web annotations are converted into properties of that instance. Several works have proposed using controlled natural language to enter OWL constructs as a way of simplifying construction of OWL ontologies [19,20]. These tools make a variety of trade-offs in terms of which constructs to present to the users. To the best of our knowledge, WebProtégé is the first web-based interface for ontology development designed empirically, based on a large ontology corpus.

We used one corpus—BioPortal—to design the interface. Our evaluation of this interface against a new corpus demonstrated the general validity of our approach. At the same time, it highlights two key types of axioms—disjoint classes axioms

and equivalent classes axioms—that account for a notable fraction of this new corpus that cannot be represented in the WebProtégé profile. We are currently evaluating several approaches to extend the expressive power of the user interface. First, we can expand the default UI to account for these types of axioms. We will evaluate how much it affects the simplicity and usability of the interface: there is a danger that adding more expressive power will clutter the interface and take away what the users currently like about it. Second, we can design a second preconfigured interface, which will be geared towards the users who are more experienced with OWL and will provide greater expressive power. We plan to investigate whether or not we can limit the default interface to a single interface that satisfies all our users (something that our survey indicates might be possible) or whether we need multiple configurations. Finally, we can leave it up to the users to configure the WebProtégé UI to satisfy their needs. One of the key features of WebProtégé is that it allows users to custom-tailor their interface, choosing which components they see in the class definitions, and which widgets they use from each component. For instance, a user that needs to write complex OWL class expressions that are not supported by the simple UI can enable a UI component that looks similar to the class description editor in the desktop version of Protégé.

8 Conclusions

In this paper, we presented the new version of WebProtégé, a web-based OWL ontology editor with an empirically designed simple user interface. This user interface accounts for a large fraction of ontologies and class frames in two large ontology corpora. Yet, a mix of beginner and expert users perceive it as being both easy to use and easy to learn and they are satisfied with the interface. Our data shows a significant community uptake. These results point to a novel way to address the complexity of ontology development through an iterative process that relies on empirical data and feedback from the user community.

Acknowledgements. This work was supported by grants GM086587 and GM103316 from the National Institute of General Medical Sciences at the United States National Institute of Health. We are indebted to all the Protégé users for their continuous feedback and support.

References

1. Gibson, A., Wolstencroft, K., Stevens, R.: Promotion of ontological comprehension: Exposing terms and metadata with Web 2.0. In: Workshop on Social and Collaborative Construction of Structured Knowledge at WWW 2007 (2007)
2. Nixon, L., García-Castro, R., Wrigley, S., Yatskevich, M., Santos, C.T.D., Cabral, L.: The state of semantic technology today - overview of the first SEALS evaluation campaigns. In: 7th Int. Conf. on Semantic Systems (2011)
3. Horridge, M., Bechhofer, S.: The OWL API: A Java API for OWL ontologies. Semantic Web Journal 2(1), 11–21 (2011)
4. Musen, M.A., Noy, N.F., Shah, N.H., Whetzel, P.L., Chute, C.G., Storey, M.A., Smith, B.: The NCBO team: The National Center for Biomedical Ontology. Journal of American Medical Informatics Association 19, 190–195 (2012)

5. Whetzel, P.L., Noy, N.F., Shah, N.H., Alexander, P.R., Nyulas, C.I., Tudorache, T., Musen, M.A.: BioPortal: Enhanced functionality via new web services from the national center for biomedical ontology to access and use ontologies in software applications. Nucleic Acids Research (NAR) 39(Web Server issue), W541–W545 (2011)
6. Motik, B., Patel-Schneider, P.F., Parsia, B.: OWL 2 Web Ontology Language structural specification and functional style syntax. W3C Recommendation. In: W3C – World Wide Web Consortium (2009)
7. Mungall, C.: OBO Flat File Format 1.4 syntax and semantics (2011)
8. Golbreich, C., Horridge, M., Horrocks, I., Motik, B., Shearer, R.: OBO and OWL: Leveraging semantic web technologies for the life sciences. In: Aberer, K., Choi, K.-S., Noy, N., Allemang, D., Lee, K.-I., Nixon, L.J.B., Golbeck, J., Mika, P., Maynard, D., Mizoguchi, R., Schreiber, G., Cudré-Mauroux, P. (eds.) ASWC 2007 and ISWC 2007. LNCS, vol. 4825, pp. 169–182. Springer, Heidelberg (2007)
9. Tudorache, T., Nyulas, C., Noy, N.F., Musen, M.A.: WebProtégé: A collaborative ontology editor and knowledge acquisition tool for the web. Semantic Web Journal 4(1) (2013)
10. Corcho, Ó., Roussey, C.: OnlynessIsLoneliness (oil). In: Proceedings of the Workshop on Ontology Patterns (WOP 2009) (2009)
11. Rector, A.L., Drummond, N., Horridge, M., Rogers, J.D., Knublauch, H., Stevens, R., Wang, H., Wroe, C.: OWL pizzas: Practical experience of teaching OWL-DL: Common errors & common patterns. In: Motta, E., Shadbolt, N.R., Stutt, A., Gibbins, N. (eds.) EKAW 2004. LNCS (LNAI), vol. 3257, pp. 63–81. Springer, Heidelberg (2004)
12. Horridge, M., Parsia, B., Sattler, U.: The state of biomedical ontologies. In: BioOntologies 2011 Co-Located with ISMB (2011)
13. Nielsen, J.: Usability Engineering. Academic Press/Morgan Kaufmann (1994)
14. Auer, S., Dietzold, S., Riechert, T.: OntoWiki–a tool for social, semantic collaboration. In: Cruz, I., Decker, S., Allemang, D., Preist, C., Schwabe, D., Mika, P., Uschold, M., Aroyo, L.M. (eds.) ISWC 2006. LNCS, vol. 4273, pp. 736–749. Springer, Heidelberg (2006)
15. Ghidini, C., Kump, B., Lindstaedt, S., Mahbub, N., Pammer, V., Rospocher, M., Serafini, L.: MoKi: The enterprise modelling wiki. In: Aroyo, L., Traverso, P., Ciravegna, F., Cimiano, P., Heath, T., Hyvönen, E., Mizoguchi, R., Oren, E., Sabou, M., Simperl, E. (eds.) ESWC 2009. LNCS, vol. 5554, pp. 831–835. Springer, Heidelberg (2009)
16. Basca, C., Corlosquet, S., Cyganiak, R., Fernández, S., Schandl, T.: Neologism: Easy vocabulary publishing. In: Workshop on Scripting for the Semantic Web (ESWC 2008) (2008)
17. Schandl, T., Blumauer, A.: Poolparty: Skos thesaurus management utilizing linked data. In: Aroyo, L., Antoniou, G., Hyvönen, E., ten Teije, A., Stuckenschmidt, H., Cabral, L., Tudorache, T. (eds.) ESWC 2010, Part II. LNCS, vol. 6089, pp. 421–425. Springer, Heidelberg (2010)
18. Krötzsch, M., Vrandečić, D., Völkel, M.: Semantic MediaWiki. In: Cruz, I., Decker, S., Allemang, D., Preist, C., Schwabe, D., Mika, P., Uschold, M., Aroyo, L.M. (eds.) ISWC 2006. LNCS, vol. 4273, pp. 935–942. Springer, Heidelberg (2006)
19. Hart, G., Johnson, M., Dolbear, C.: Rabbit: Developing a control natural language for authoring ontologies. In: Bechhofer, S., Hauswirth, M., Hoffmann, J., Koubarakis, M. (eds.) ESWC 2008. LNCS, vol. 5021, pp. 348–360. Springer, Heidelberg (2008)
20. Power, R.: OWL simplified english: A finite-state language for ontology editing. In: Kuhn, T., Fuchs, N.E. (eds.) CNL 2012. LNCS, vol. 7427, pp. 44–60. Springer, Heidelberg (2012)

A Query Tool for $\mathcal{EL}$ with Non-monotonic Rules

Vadim Ivanov[1,2], Matthias Knorr[1], and João Leite[1]

[1] CENTRIA & Departamento de Informática, Universidade Nova de Lisboa, Portugal
[2] Department of Computing Mathematics and Cybernetics,
Ufa State Aviation Technical University, Russia

Abstract. We present the Protégé plug-in NoHR that allows the user to take an $\mathcal{EL}^+_\bot$ ontology, add a set of non-monotonic (logic programming) rules – suitable e.g. to express defaults and exceptions – and query the combined knowledge base. Our approach uses the well-founded semantics for MKNF knowledge bases as underlying formalism, so no restriction other than DL-safety is imposed on the rules that can be written. The tool itself builds on the procedure **SLG**($\mathcal{O}$) and, with the help of OWL 2 EL reasoner ELK, pre-processes the ontology into rules, whose result together with the non-monotonic rules serve as input for the top-down querying engine XSB Prolog. With the resulting plug-in, even queries to very large ontologies, such as SNOMED CT, augmented with a large number of rules, can be processed at an interactive response time after one initial brief pre-processing period. At the same time, our system is able to deal with possible inconsistencies between the rules and an ontology that alone is consistent.

1 Introduction

Ontology languages have become widely used to represent and reason over taxonomic knowledge, and often such knowledge bases are expressed within the language of the OWL 2 profile OWL 2 EL [23]. For example, the clinical health care terminology SNOMED CT,[1] arguably the most prominent example in the area of medicine and currently used for electronic health record systems, clinical decision support systems, or remote intensive care monitoring, to name only a few, builds on a fragment of OWL 2 EL and its underlying description logic (DL) $\mathcal{EL}^{++}$ [5].

Whereas the OWL ontology languages based on DLs [4] are monotonic by nature, which means that once drawn conclusions persist when adopting new additional information, the ability to model defaults and exceptions with a closed-world view is frequently requested as a missing feature. For example, in [25], modeling pharmacy data of patients with the closed-world assumption would have been preferred in the study to match patient records with clinical trials criteria, because usually it can be assumed that a patient is not under a specific medication unless explicitly known. Similarly, in clinical health care terminology, it would be advantageous to be able to express that normally the heart is on the left side of the body unless the person is a dextrocardiac.

In recent years, there has been a considerable amount of effort devoted to extending DLs with non-monotonic features – see, e.g., related work in [12,24]) – and many of the existing approaches focus on combining DLs and non-monotonic rules as known

[1] `http://www.ihtsdo.org/snomed-ct/`

H. Alani et al. (Eds.): ISWC 2013, Part I, LNCS 8218, pp. 216–231, 2013.

from Logic Programming. The latter are one of the most well-studied formalisms that admit expressing defaults, exceptions, and also integrity constraints in a declarative way and are part of RIF [6], the other expressive language for the Semantic Web whose standardization is driven by the W3C.[2]

Here we focus on one such combination – Hybrid MKNF under the well-founded semantics [20] – for two reasons. First, the overall approach, which was introduced in [24] and is based on the logic of minimal knowledge and negation as failure (MKNF) [22], provides a very general and flexible framework for combining DL ontologies and non-monotonic rules (see [24]). Second, [20], which is a variant of [24] based on the well-founded semantics [14] for logic programs, has a (lower) polynomial data complexity and is amenable for applying top-down query procedures, such as **SLG**($\mathcal{O}$) [2], to answer queries based only on the information relevant for the query, and without computing the entire model – no doubt a crucial feature when dealing with large ontologies such as SNOMED with over 300,000 classes or [25] with millions of assertions.

In this paper, we describe the system NoHR, realized as a plug-in for the ontology editor Protégé 4.X,[3] that allows the user to query combinations of $\mathcal{EL}^+_\bot$ ontologies and non-monotonic rules in a top-down manner. To the best of our knowledge, it is the first Protégé plug-in to integrate non-monotonic rules and top-down queries. Our approach is theoretically founded in the abstract procedure **SLG**($\mathcal{O}$) and based on utilizing the consequence-driven, concurrent $\mathcal{EL}$ reasoner ELK, which is considerably faster than other $\mathcal{EL}$ reasoners [18], to classify the ontology part and then translate the result into rules which, together with the non-monotonic rules, serve as input for the top-down query engine XSB Prolog.[4] Additional features of the plug-in include: the possibility to load and edit rule bases, and define predicates with arbitrary arity; guaranteed termination of query answering, with a choice between one/many answers; robustness w.r.t. inconsistencies between the ontology and the rule part. Our main contributions are:

- We generalize the procedure presented in [2] to avoid the normalization of $\mathcal{EL}^+_\bot$ knowledge bases, which is not necessary for ELK, also reducing the size of the XSB file and of the tables in XSB. At the same time we significantly improve the formalization in [2] – including the correct handling of complex concept assertions – in order to show that our procedure is correct.
- We describe an implementation of this revised procedure in Java including an ELK reasoner for preprocessing the ontology, whose translated output, together with the rules, can be queried interactively under XSB via the Java front-end Interprolog.[5]
- We evaluate the performance of our tool, showing that even SNOMED augmented with a large number of rules can be preprocessed in a brief period of time and then query answering is possible at interactive response time.

The remainder of the paper is structured as follows. In Sect. 2, we briefly recall the DL $\mathcal{EL}^+_\bot$ and MKNF knowledge bases as a tight combination of the former DL and non-monotonic rules. Then, we present the revised reasoning algorithm that allows us

[2] `http://www.w3.org`
[3] `http://protege.stanford.edu`
[4] `http://xsb.sourceforge.net`
[5] `http://www.declarativa.com/interprolog/`

Table 1. Syntax and semantics of $\mathcal{EL}^+_\bot$

	Syntax	Semantics
atomic concept	$A \in \mathsf{N_C}$	$A^\mathcal{I} \subseteq \Delta^\mathcal{I}$
atomic role	$R \in \mathsf{N_R}$	$R^\mathcal{I} \subseteq \Delta^\mathcal{I} \times \Delta^\mathcal{I}$
individual	$a \in \mathsf{N_I}$	$a^\mathcal{I} \in \Delta^\mathcal{I}$
top	$\top$	$\Delta^\mathcal{I}$
bottom	$\bot$	$\emptyset$
conjunction	$C \sqcap D$	$C^\mathcal{I} \cap D^\mathcal{I}$
existential restriction	$\exists R.C$	$\{x \in \Delta^\mathcal{I} \mid \exists y \in \Delta^\mathcal{I} : (x,y) \in R^\mathcal{I} \wedge y \in C^\mathcal{I}\}$
concept inclusion	$C \sqsubseteq D$	$C^\mathcal{I} \subseteq D^\mathcal{I}$
role inclusion	$R \sqsubseteq S$	$R^\mathcal{I} \subseteq S^\mathcal{I}$
role composition	$R_1 \circ \cdots \circ R_k \sqsubseteq S$	$(x_1,x_2) \in R_1^\mathcal{I} \wedge \ldots \wedge (x_k,y) \in R_k^\mathcal{I} \rightarrow (x_1,y) \in S^\mathcal{I}$
concept assertion	$C(a)$	$a^\mathcal{I} \in C^\mathcal{I}$
role assertion	$R(a,b)$	$(a^\mathcal{I}, b^\mathcal{I}) \in R^\mathcal{I}$

to query such MKNF knowledge bases in Sect. 3. In Sect. 4, we introduce our implementation of the plug-in and evaluate it in Sect. 5, before we conclude in Sect. 6.

2 Preliminaries

2.1 Description Logic $\mathcal{EL}^+_\bot$

We start by recalling the syntax and semantics of $\mathcal{EL}^+_\bot$, a large fragment of $\mathcal{EL}^{++}$ [5], the DL underlying the tractable profile OWL 2 EL [23], following the presentation in [18,19]. For a more general and thorough introduction to DLs we refer to [4].

The language of $\mathcal{EL}^+_\bot$ is defined over countably infinite sets of *concept names* $\mathsf{N_C}$, *role names* $\mathsf{N_R}$, and *individual names* $\mathsf{N_I}$ as shown in the upper part of Table 1. Building on these, *complex concepts* are introduced in the middle part of Table 1, which, together with atomic concepts, form the set of *concepts*. We conveniently denote individuals by a and b, (atomic) roles by R and S, atomic concepts by A and B, and concepts by C and D. All expressions in the lower part of Table 1 are *axioms*. A *concept equivalence* $C \equiv D$ is an abbreviation for $C \sqsubseteq D$ and $D \sqsubseteq C$. Concept and role assertions are *ABox axioms* and all other axioms *TBox axioms*, and an *ontology* is a finite set of axioms.

The semantics of $\mathcal{EL}^+_\bot$ is defined in terms of an *interpretation* $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$ consisting of a non-empty domain $\Delta^\mathcal{I}$ and an *interpretation function* $\cdot^\mathcal{I}$. The latter is defined for (arbitrary) concepts, roles, and individuals as in Table 1. Moreover, an interpretation $\mathcal{I}$ *satisfies* an axiom α, written $\mathcal{I} \models \alpha$, if the corresponding condition in Table 1 holds. If $\mathcal{I}$ satisfies all axioms occurring in an ontology $\mathcal{O}$, then $\mathcal{I}$ is a *model* of $\mathcal{O}$, written $\mathcal{I} \models \mathcal{O}$. If $\mathcal{O}$ has at least one model, then it is called *consistent*, otherwise *inconsistent*. Also, $\mathcal{O}$ *entails* axiom α, written $\mathcal{O} \models \alpha$, if every model of $\mathcal{O}$ satisfies α. *Classification* requires to compute all concept inclusions between atomic concepts entailed by $\mathcal{O}$.

2.2 MKNF Knowledge Bases

MKNF knowledge bases (KBs) build on the logic of minimal knowledge and negation as failure (MKNF) [22]. Two main different semantics have been defined [24,20], and we focus on the well-founded version [20], due to its lower computational complexity and amenability to top-down querying without computing the entire model. Here, we only point out important notions, and refer to [20] and [2] for the details.

We start by recalling MKNF knowledge bases as presented in [2] to combine an ($\mathcal{EL}^+_\bot$) ontology and a set of non-monotonic rules (similar to a normal logic program).

Definition 1. *Let $\mathcal{O}$ be an ontology. A function-free first-order* atom $P(t_1, \ldots, t_n)$ *such that P occurs in $\mathcal{O}$ is called* DL-atom*; otherwise it is called* non-DL-atom. *A* rule r *is of the form*

$$H \leftarrow A_1, \ldots, A_n, \mathbf{not} B_1, \ldots, \mathbf{not} B_m \tag{1}$$

where the head *of r, H, and all A_i with $1 \leq i \leq n$ and B_j with $1 \leq j \leq m$ in the* body *of r are atoms. A* program $\mathcal{P}$ *is a finite set of rules, and an* MKNF knowledge base $\mathcal{K}$ *is a pair $(\mathcal{O}, \mathcal{P})$. A rule r is* DL-safe *if all its variables occur in at least one non-DL-atom A_i with $1 \leq i \leq n$, and $\mathcal{K}$ is* DL-safe *if all its rules are DL-safe.*

DL-safety ensures decidability of reasoning with MKNF knowledge bases and can be achieved by introducing a new predicate o, adding $o(i)$ to $\mathcal{P}$ for all constants i appearing in $\mathcal{K}$ and, for each rule $r \in \mathcal{P}$, adding $o(X)$ for each variable X appearing in r to the body of r. Therefore, we only consider DL-safe MKNF knowledge bases.

Example 2. Consider an MKNF knowledge base for recommending vacation destinations taken from [24] (with a few modifications). We denote DL-atoms and constants with upper-case names and non-DL-atoms and variables with lower-case names.[6]

$$\begin{aligned}
PortCity(Barcelona) \quad & OnSea(Barcelona, Mediterranean)\\
PortCity(Hamburg) \quad & NonSeaSideCity(Hamburg)\\
RainyCity(Manchester) \quad & Has(Manchester, AquaticsCenter)\\
Recreational(AquaticsCenter) &\\
SeaSideCity \sqsubseteq \exists Has.Beach &\\
Beach \sqsubseteq Recreational &\\
\exists Has.Recreational \sqsubseteq RecreationalCity &\\
SeaSideCity(x) \leftarrow PortCity(x), \mathbf{not}\, NonSeaSideCity(x) &\\
interestingCity(x) \leftarrow RecreationalCity(x), \mathbf{not}\, RainyCity(x) &\\
hasOnSea(x) \leftarrow OnSea(x, y) &\\
false \leftarrow SeaSideCity(x), \mathbf{not}\, hasOnSea(x) &\\
summerDestination(x) \leftarrow interestingCity(x), OnSea(x, y) &
\end{aligned}$$

This example shows that we can seamlessly express defaults and exceptions, such as every port city normally being a seaside city, integrity constraints, such as requiring to know for every seaside city on which sea it lies, and at the same time taxonomic/ontological knowledge including information over unknown individuals, such as

[6] To ease readability, we omit the auxiliary atoms that ensure DL-safety and leave them implicit.

a seaside city being recreational even if we do not know the specific name of the beach. Note that, unlike [24], the rule with head $false$ is not a true integrity constraint in our case. Rather, whenever the keyword $false$ would be derivable, we know that there is at least one seaside city for which we do not know on which sea it lies.

The semantics of MKNF knowledge bases $\mathcal{K}$ is usually given by a translation π into an MKNF formula $\pi(\mathcal{K})$, i.e., a formula over first-order logic extended with two modal operators $\mathbf{K}$ and $\mathbf{not}$. Namely, every rule of the form (1) is translated into $\mathbf{K}H \leftarrow \mathbf{K}A_1, \ldots, \mathbf{K}A_n, \mathbf{not}B_1, \ldots, \mathbf{not}B_m$, $\pi(\mathcal{P})$ is the conjunction of the translations of its rules, and $\pi(\mathcal{K}) = \mathbf{K}\pi(\mathcal{O}) \wedge \pi(\mathcal{P})$ where $\pi(\mathcal{O})$ is the first-order translation of $\mathcal{O}$. Reasoning with such MKNF formulas is then commonly achieved using a partition of *modal atoms*, i.e., all expressions of the form $\mathbf{K}\varphi$ for each $\mathbf{K}\varphi$ or $\mathbf{not}\varphi$ occurring in $\pi(\mathcal{K})$. For [20], such a partition assigns *true*, *false*, or *undefined* to (modal) atoms, and can be effectively computed in polynomial time. If $\mathcal{K}$ is *MKNF-consistent*, then this partition does correspond to the unique model of $\mathcal{K}$ [20], and, like in [2], we call the partition the *well-founded MKNF model* $\mathsf{M}_{\mathsf{wf}}(\mathcal{K})$. Here, $\mathcal{K}$ may indeed not be MKNF-consistent if the $\mathcal{EL}^+_\bot$ ontology alone is inconsistent, which is possible if $\bot$ occurs, or by the combination of appropriate axioms in $\mathcal{O}$ and $\mathcal{P}$, e.g., $A \sqsubseteq \bot$ and $A(a) \leftarrow$. In the former case, we argue that the ontology alone should be consistent and be repaired if necessary before combining it with non-monotonic rules. Thus, we assume in the following that $\mathcal{O}$ occurring in $\mathcal{K}$ is consistent.

2.3 Querying in MKNF Knowledge Bases

In [2], a procedure, called **SLG($\mathcal{O}$)**, is defined for querying MKNF knowledge bases under the well-founded MKNF semantics. This procedure extends SLG resolution with tabling [9] with an *oracle* to $\mathcal{O}$ that handles ground queries to the DL-part of $\mathcal{K}$ by returning (possibly empty) sets of atoms that, together with $\mathcal{O}$ and information already proven true, allows us to derive the queried atom. We refer to [2] for the full account of **SLG($\mathcal{O}$)**, and only recall a few crucial notions necessary in the following.

SLG($\mathcal{O}$) is based on creating top-down derivation trees with the aim of answering *(DL-safe) conjunctive queries* Q of the form $q(\boldsymbol{X}) \leftarrow A_1, \ldots, A_n, \mathbf{not}B_1, \ldots, \mathbf{not}B_m$ where each variable in Q occurs in at least one non-DL atom in Q, and where $\boldsymbol{X}$ is the (possibly empty) set of requested variables appearing in the body.

In general, the computation of $\mathsf{M}_{\mathsf{wf}}(\mathcal{K})$ uses two different versions of $\mathcal{K}$ in parallel to guarantee that a) coherence is ensured, i.e., if $\neg P(a)$ is derivable, then $\mathbf{not}P(a)$ has to be true as well (cf. also [20]), and b) MKNF-consistency of $\mathcal{K}$ can be verified. For a top-down approach this is impractical, so, instead, a doubled MKNF knowledge base $\mathcal{K}^d = (\mathcal{O}, \mathcal{O}^d, \mathcal{P}^d)$ is defined in which a copy of $\mathcal{O}$ with new doubled predicates is added, and two rules occur in $\mathcal{P}^d$ for each rule in $\mathcal{P}$, intertwining original and doubled predicates (see Def. 3.1 in [2]). It is shown that an atom A is true in $\mathsf{M}_{\mathsf{wf}}(\mathcal{K})$ iff A is true in $\mathsf{M}_{\mathsf{wf}}(\mathcal{K}^d)$ and A is false in $\mathsf{M}_{\mathsf{wf}}(\mathcal{K})$ iff A^d is false in $\mathsf{M}_{\mathsf{wf}}(\mathcal{K}^d)$. Note that $\mathcal{K}^d$ is indeed necessary in general, but if $\mathcal{K}$ does not contain $\bot$, then we can use $\mathcal{K}$ directly.

In [2], the notion of oracle is defined to handle ground queries to the ontology, but before we recall that notion, we use an example to illustrate the idea.

Example 3. Recall $\mathcal{K}$ in Ex. 2. Since $\bot$ does not occur in $\mathcal{K}$, we can restrict ourselves to $\mathcal{K}$ here. First, consider query $q = interestingCity(Manchester)$. We find a rule whose head unifies with q, and obtain two new queries, $RecreationalCity(Manchester)$ and $\mathbf{not}\, RainyCity(Manchester)$. There is no rule whose head matches the former, but we can query the ontology and the answer is yes together with an empty set of atoms, i.e., $RecreationalCity(Manchester)$ can be proven from $\mathcal{O}$ alone. Now we handle $\mathbf{not}\, RainyCity(Manchester)$, so we query $RainyCity(Manchester)$ which can also be proven by $\mathcal{O}$ alone. Therefore $\mathbf{not}\, RainyCity(Manchester)$ fails, so q is false.

Now, consider $q_1 = interestingCity(Barcelona)$. We obtain again two new queries, $q_2 = RecreationalCity(Barcelona)$ and $q_3 = \mathbf{not}\, RainyCity(Barcelona)$. In this case, $q_2 = RecreationalCity(Barcelona)$ cannot be proven from $\mathcal{O}$ alone, but the oracle could return $Has(Barcelona, X)$ and $Recreational(X)$, which, if we would find a value for X, would allow us to derive q_2. However, neither of the two atoms appear in a rule head in $\mathcal{P}$, so we will never be able to derive it from $\mathcal{P}$. In fact, the only proper answer the oracle may return is $q_4 = SeaSideCity(Barcelona)$. From the corresponding rule in $\mathcal{P}$ we obtain two new queries $q_5 = PortCity(Barcelona)$ and $q_6 = \mathbf{not}\, NonSeaSideCity(Barcelona)$. Then, q_5 can be derived from $\mathcal{O}$ alone, and q_6 succeeds, because $NonSeaSideCity(Barcelona)$ fails. So q_4 succeeds, and therefore also q_2. Finally q_3 succeeds since $RainyCity(Barcelona)$ fails, so q_1 is true.

We recall the notions of a complete and a (correct) partial oracle from [2].

Definition 4. *Let $\mathcal{K}^d = (\mathcal{O}, \mathcal{O}^d, \mathcal{P}^d)$ be a doubled MKNF KB, $\mathcal{I}$ a set of ground atoms (already proven to be true), S a ground query, and $\mathcal{L}$ a set of ground atoms such that each $L \in \mathcal{L}$ is unifiable with at least one rule head in $\mathcal{P}^d$. The* complete oracle *for $\mathcal{O}$, denoted $compT_\mathcal{O}$, is defined by $compT_\mathcal{O}(\mathcal{I}, S, \mathcal{L})$ iff $\mathcal{O} \cup \mathcal{I} \cup \mathcal{L} \models S$ or $\mathcal{O}^d \cup \mathcal{I} \cup \mathcal{L} \models S$. A* partial oracle *for $\mathcal{O}$, denoted $pT_\mathcal{O}$, is a relation $pT_\mathcal{O}(\mathcal{I}, S, \mathcal{L})$ such that if $pT_\mathcal{O}(\mathcal{I}, S, \mathcal{L})$, then $\mathcal{O} \cup \mathcal{I} \cup \mathcal{L} \models S$ or $\mathcal{O}^d \cup \mathcal{I} \cup \mathcal{L} \models S$ for consistent $\mathcal{O} \cup \mathcal{I} \cup \mathcal{L}$ and $\mathcal{O}^d \cup \mathcal{I} \cup \mathcal{L}$, respectively.*

A partial oracle $pT_\mathcal{O}$ is correct *w.r.t. $compT_\mathcal{O}$ iff, for all MKNF-consistent $\mathcal{K}^d$, replacing $compT_\mathcal{O}$ in* **SLG($\mathcal{O}$)** *with $pT_\mathcal{O}$ succeeds for exactly the same set of queries.*

Partial oracles may avoid returning unnecessary answers $\mathcal{L}$, such as non-minimal answers or those that try to derive an MKNF-inconsistency even though $\mathcal{K}^d$ is MKNF-consistent. Moreover, correctness of partial oracles is only defined w.r.t MKNF-consistent $\mathcal{K}$. The rationale is that, when querying top-down, we want to avoid checking whether the entire KB $\mathcal{K}^d$ is MKNF-consistent. This leads to para-consistent derivations if $\mathcal{K}^d$ is not MKNF-consistent, e.g., some atom P is true, yet P^d is false, while other independent atoms are evaluated as if $\mathcal{K}^d$ was MKNF-consistent (see [2]).

3 Pre-processing the Ontology and Querying

In [2], an **SLG($\mathcal{O}$)** oracle for $\mathcal{EL}^+_\bot$ is defined based on the reasoning algorithm for ontology classification presented in [5], which is restricted to normalized ontologies.

Even though the process of normalizing $\mathcal{EL}$ ontologies is linear, it introduces auxiliary predicates to achieve the normal form, which, e.g., not only is counter-intuitive to the idea of finding meaningful explanations for information derived by top-down queries but also increases the size of the resulting XSB file and tables used in XSB. Since the reasoning algorithm of ELK [18,19] is defined for the general case, and ELK is the reasoner we want to use for implementing our query-tool, because it is considerably faster than other $\mathcal{EL}$ reasoners, we generalize the algorithm in [2] to non-normalized ontologies. At the same time, we fix a problem in [2] w.r.t. handling non-atomic concept assertions, and we improve the formalization to prove correctness of our oracle.

In the following, we first utilize the algorithm underlying ELK to compute implicit information derivable from a given ontology. After that, we discard certain axioms, because, with the implicit information computed, they are no longer required for the query task used in **SLG(**$\mathcal{O}$**)**. Then, we translate the remaining set of axioms into rules, which can equally be used as an $\mathcal{EL}$ oracle.

3.1 Simplifying the Ontology

The basic idea for an $\mathcal{EL}^+_\bot$ oracle is to translate the ontology into rules. The only obstacle are concept inclusions with $\exists R.C$ on the right-hand side since these cannot be translated straightforwardly. However, such axioms alone can never contribute to oracle derivations. Consider, e.g., querying $\mathcal{K}$ in Ex. 2 for $Beach(i)$ for any constant i: an oracle cannot derive $Beach(i)$ even if $SeaSideCity(c)$ and $has(c, i)$ were already known to be true. Yet, such axioms are useful indirectly, e.g., to obtain that proving $SeaSideCity(Barcelona)$ would suffice to derive $RecreationalCity(Barcelona)$. So, classification is applied first to make such implicit links explicit, and then all concept inclusions with sub-concepts $\exists R.C$ on the right-hand side can be rewritten or removed.

The consequence-based procedure for classification of TBoxes in $\mathcal{EL}^+_\bot$ is described in [19]. Here we only sketch it with a focus on the results important for our purposes.

First, the initial set $\mathbf{input}$ is defined to contain one axiom $\mathbf{init}(A)$ for each atomic concept A in $\mathcal{O}$. Then, a set of $\mathcal{EL}^+_\bot$ *inference rules* for TBox reasoning in $\mathcal{EL}^+_\bot$ (see [19]) is applied exhaustively to $\mathbf{input}$, yielding $\mathbf{Closure}$ as final result, which contains axioms derivable from $\mathcal{O}$ and axioms of the form $\mathbf{init}(C)$ and $C \xrightarrow{\mathrm{R}} D$, where the latter represents that, for two (initialized) concepts C and D, $C \sqsubseteq \exists R.D$ is entailed.

Theorem 5 ([19]). *Let $\mathcal{O}$ be an $\mathcal{EL}^+_\bot$ ontology,* $\mathbf{input}$ *a set of expressions* $\mathbf{init}(C)$*, and* $\mathbf{Closure}$ *the closure of* $\mathbf{input}$ *under the $\mathcal{EL}^+_\bot$ inference rules w.r.t. $\mathcal{O}$. Then, for each concept C such that* $\mathbf{init}(C) \in \mathbf{Closure}$ *and each atomic concept A, we have*

1. $\mathcal{O} \models C \sqsubseteq \bot$ *iff* $C \sqsubseteq \bot \in \mathbf{Closure}$*,*
2. $\mathcal{O} \models C \sqsubseteq A$ *iff* $C \sqsubseteq \bot \in \mathbf{Closure}$ *or* $C \sqsubseteq A \in \mathbf{Closure}$*.*

Note that one of the inference rules ($\mathsf{R}^-_\exists$) allows $\mathbf{init}(D) \in \mathbf{Closure}$ if $C \sqsubseteq \exists R.D \in \mathbf{Closure}$. Hence, Theorem 5 does not apply only to $\mathbf{init}(C) \in \mathbf{input}$.

One thing missing so far, which is not correctly covered in the translation presented in [2], is that concept assertions also have to be considered since, e.g., $\exists R.C(a)$ together

with $\exists R.C \sqsubseteq D$, make $D(a)$ derivable, yet the $\mathcal{EL}^+_\bot$ inference rules are defined for TBox axioms. The solution, adapted from [19], is to apply a transformation N that translates concept assertions $C(a)$ into concept inclusions $N_a \sqsubseteq C$, where the set of atomic concepts contains an atomic concept N_a for every $a \in \mathsf{N_I}$ s.t. N_a does not appear in $\mathcal{O}$, and leaves all other axioms unchanged. Note that we do not translate role assertions $R(a, b)$, because R is just an atomic role without occurrences of concepts of the form $\exists R.C$. Still, it holds for consistent $\mathcal{O}$ and axiom α that do not contain atomic concepts of the form N_a, that $\mathcal{O} \models \alpha$ iff $N(\mathcal{O}) \models N(\alpha)$ ([19], Theorem 3).

We are now ready to present the new definition of a reduced ontology.

Definition 6. *Let $\mathcal{O}$ be an $\mathcal{EL}^+_\bot$ ontology,* $\mathbf{input}$ *a set of expressions* $\mathbf{init}(A)$ *for each atomic concept A, and* $\mathbf{Closure}$ *the closure of* $\mathbf{input}$ *under the $\mathcal{EL}^+_\bot$ inference rules w.r.t. $N(\mathcal{O})$. The* reduced ontology *of $\mathcal{O}$ is obtained from $\mathcal{O}_1 = N(\mathcal{O}) \cup \mathbf{Closure}$ as follows.*

1. *Remove all statements of the forms $\mathbf{init}(C)$, $C \xrightarrow{R} D$, and $C \sqsubseteq \exists R.D$ from $\mathcal{O}_1$;*
2. *Remove all sub-concepts of the form $\exists R.D$ from the right-hand side of any axiom $C \sqsubseteq D_1 \sqcap D_2 \in \mathcal{O}_1$; if no conjunct is left, remove the entire axiom from $\mathcal{O}_1$;*
3. *Substitute all concept inclusions of the form $N_a \sqsubseteq C$ remaining after 1. and 2. by $C(a)$ for all $a \in \mathsf{N_I}$.*

Note that $\mathbf{input}$ does contain $\mathbf{init}(N_a)$ for all $a \in \mathsf{N_I}$. Moreover, steps 1. and 2. already remove all sub-concepts of the form $\exists R.D$ from concept inclusions that represent concept assertions, so indeed the reduced ontology does not contain concepts of the form $\exists R.D$ in concept assertions and the right-hand sides of concept inclusions. We can show that reduced $\mathcal{O}$ contain all atomic concept assertions they entail.

Lemma 7. *Let $\mathcal{O}$ be reduced and A an atomic concept. If $\mathcal{O} \models A(a)$, then $A(a) \in \mathcal{O}$.*

Also $\mathcal{O}$ and the reduced $\mathcal{O}'$ entail the same unary and binary atoms.

Lemma 8. *Let $\mathcal{O}$ be an $\mathcal{EL}^+_\bot$ ontology, $\mathcal{O}'$ the reduced ontology of $\mathcal{O}$, A a unary and R a binary predicate: $\mathcal{O} \models A(a)$ iff $\mathcal{O}' \models A(a)$ and $\mathcal{O} \models R(a, b)$ iff $\mathcal{O}' \models R(a, b)$.*

Now, we show that we can use the reduced ontology of $\mathcal{O}$ instead of $\mathcal{O}$ as an $\mathcal{EL}^+_\bot$ oracle. First, we specify a partial oracle that is necessarily a correct partial oracle for $\mathcal{O}$.

Definition 9. *Let $\mathcal{K}^d = (\mathcal{O}, \mathcal{O}^d, \mathcal{P}^d)$ be a doubled MKNF KB, $\mathcal{I}$ a set of ground atoms (already proven to be true), S a ground query, and $\mathcal{L}$ a set of ground atoms such that each $L \in \mathcal{L}$ is unifiable with at least one rule head in $\mathcal{P}^d$. The* abstract partial oracle *for $\mathcal{O}$, denoted $pT^a_\mathcal{O}$, is a relation $pT^a_\mathcal{O}(\mathcal{I}, S, \mathcal{L})$ such that $pT^a_\mathcal{O}(\mathcal{I}, S, \mathcal{L})$ iff $\mathcal{O} \cup \mathcal{I} \cup \mathcal{L} \models S$ or $\mathcal{O}^d \cup \mathcal{I} \cup \mathcal{L} \models S$ for consistent $\mathcal{O} \cup \mathcal{I} \cup \mathcal{L}$ and $\mathcal{O}^d \cup \mathcal{I} \cup \mathcal{L}$, respectively.*

Such an abstract partial oracle is not necessarily efficient, since it does return all possible consistent tuples $(\mathcal{I}, S, \mathcal{L})$ but it certainly is correct.

Proposition 10. *Let $\mathcal{K}^d = (\mathcal{O}, \mathcal{O}^d, \mathcal{P}^d)$ be a doubled MKNF KB. The abstract partial oracle $pT^a_\mathcal{O}$ is correct w.r.t. $compT_\mathcal{O}$.*

Based on this result, we can show that $\mathcal{O}$ can be substituted with the reduced ontology of $\mathcal{O}$ and still yield a correct partial oracle.

Theorem 11. *Let $\mathcal{K}^d = (\mathcal{O}, \mathcal{O}^d, \mathcal{P}^d)$ be a doubled MKNF KB and $\mathcal{O}'$ the reduced ontology of $\mathcal{O}$. Then $pT^a_{\mathcal{O}'}$ is a correct partial oracle w.r.t. $compT_{\mathcal{O}}$.*

3.2 Translation into Rules

Now, we can show how to translate a reduced $\mathcal{EL}^+_\bot$ ontology into rules. The only thing missing is pointing out a technical detail on how coherence and detection of MKNF-inconsistencies is achieved in [20,2]. We would like to remind that coherence intuitively ensures that an atom being "classically" false also is false by default in the non-monotonic rules. That allows us, e.g., to derive from $C \sqsubseteq \bot$, $C(a) \leftarrow \mathbf{not} D(a)$, and $D(a) \leftarrow \mathbf{not} C(a)$ that $C(a)$ is false and $D(a)$ is true, and is therefore useful in general, even if $\mathcal{K}^d$ is MKNF-consistent. In **SLG($\mathcal{O}$)**, special atoms $NH(\boldsymbol{t_i})$ are used to represent a query $\neg H(\boldsymbol{t_i})$ to the oracle. Care must be taken when translating an ontology containing $\bot$ to rules, so that these queries still work properly. Of course, if $\bot$ does not appear in $\mathcal{O}$, then considering $\mathcal{K}$ suffices.

To ease the presentation of the following translation, we introduce a few a priori simplifications. First, in a reduced ontology, the concepts C of concept assertions $C(a)$ and the right-hand sides D of concept inclusions $C \sqsubseteq D$ are all of the form $C_1 \sqcap \ldots \sqcap C_n$ with $n > 0$. We can separate these into an equivalent set of n axioms $C_i(a)$ and $C \sqsubseteq C_i$, respectively, for $1 \leq i \leq n$, simplifying in particular where some C_i is $\bot$ and another an atomic concept. Moreover, we assume without loss of generality that $\bot$ does not occur on the left-hand side of concept inclusions. We know that $\bot \sqcap C$ and $\exists R.\bot$ are both equivalent to $\bot$ and $\bot \sqsubseteq C$ is always true, so we do not need to care translating such cases into rules. Finally, for $\top$, only axioms of the form $\top \sqsubseteq C$ and sub-concepts of the form $\exists R.\top$ deserve our attention, all other instances, namely $\top(a)$, $C \sqsubseteq \top$, or $C \sqcap \top \sqsubseteq D$, are irrelevant. Hence, $\top$ is not considered in assertions, the right-hand side of concept inclusions, and in conjunctions on the left-hand side of concept inclusions, all the more, since $\top$ does not appear in $\mathcal{P}$. We also omit the auxiliary DL-safe atoms.

First, we translate arbitrary concepts into a set of correctly connected atoms.

Definition 12. *Let C be an $\mathcal{EL}^+_\bot$ concept without occurrences of $\bot$, x a variable, and X a set of variables with $x \notin X$. We define $tr(C, x)$ as follows:*

$$tr(C,x) = \begin{cases} \{A(x)\} & \text{if } C = A \\ \emptyset & \text{if } C = \top \\ tr(C_1,x) \cup tr(C_2,x) & \text{if } C = C_1 \sqcap C_2 \\ \{R(x,y)\} \cup tr(D,y) & \text{if } C = \exists R.D \end{cases}$$

where $y \in X$ is a new variable that has not been used before. We obtain $tr(C,x)^d$ from $tr(C,x)$ by substituting all predicates P in $tr(C,x)$ with P^d, and, given a set of atoms S, $\overline{S}$ is a sequence of all atoms contained in S separated by ",".

Each y is indeed intended to be a variable that is globally new in the process of translating C. E.g., $\exists R.((\exists S.\top) \sqcap (A \sqcap \exists R.B))$ translates into $S = \{R(x, y_1), S(y_1, y_2), A(y_1),$

$R(y_1, y_3), B(y_3)\}$. Then $\overline{S}$ is simply "$R(x, y_1), S(y_1, y_2), A(y_1), R(y_1, y_3), B(y_3)$" which can appear as such in a rule body.

Definition 13. *Let $\mathcal{K} = (\mathcal{O}, \mathcal{P})$ be an MKNF KB with consistent and reduced $\mathcal{O}$. We define $\mathcal{P}^d_\mathcal{O}$ from $\mathcal{O}$, where A is an atomic concept, C a concept, D a non-atomic concept, R, R_i, S roles, and a, b individuals, as the smallest set containing:*

(a1) *for each $A(a) \in \mathcal{O}$: $A(a) \leftarrow$ and $A^d(a) \leftarrow \mathbf{not} NA(a)$.*
(a2) *for each $R(a, b) \in \mathcal{O}$: $R(a, b) \leftarrow$ and $R^d(a, b) \leftarrow \mathbf{not} NR(a, b)$.*
(t1) *for each $\top \sqsubseteq A \in \mathcal{O}$: $A(x) \leftarrow$ and $A^d(x) \leftarrow \mathbf{not} NA(x)$.*
(c1) *for each $C \sqsubseteq A \in \mathcal{O} : A(x) \leftarrow \overline{tr(C, x)}$ and $A^d(x) \leftarrow \overline{tr(C, x)^d}, \mathbf{not} NA(a)$.*
(r1) *for each $R \sqsubseteq S \in \mathcal{O}$: $S(x, y) \leftarrow R(x, y)$ and $S^d(x, y) \leftarrow R^d(x, y), \mathbf{not} NS(x, y)$.*
(r2) *for each $R_1 \circ \cdots \circ R_k \sqsubseteq S \in \mathcal{O}$: $S(x_1, y) \leftarrow R_1(x_1, x_2), \ldots, R_k(x_k, y)$ and $S^d(x_1, y) \leftarrow R^d_1(x_1, x_2), \ldots, R^d_k(x_k, y), \mathbf{not} NS(x1, y)$.*
(i1) *for each $A \sqsubseteq \bot \in \mathcal{O}$: $NA(x) \leftarrow$.*
(i2) *for each $D \sqsubseteq \underline{\bot} \in \mathcal{O}$: $\{NA(y) \leftarrow \overline{tr(D, x) \setminus \{A(y)\}} \mid A(y) \in tr(D, x)\} \cup \{NR(y, z) \leftarrow \overline{tr(D, x) \setminus \{R(y, z)\}} \mid R(y, z) \in tr(D, x)\}$.*

We create in $\mathcal{P}^d_\mathcal{O}$ the rule representation for both $\mathcal{O}$ and $\mathcal{O}^d$. Again, if $\bot$ does not occur in $\mathcal{O}$, then we can skip all rules with doubled predicates, and (i1) and (i2) will not contribute anything either. The additional default atoms of the form $\mathbf{not} NA(x)$ and $\mathbf{not} NS(x, y)$ are added to the doubled rules to be in line with the idea of the doubling of rules in [2]: whenever, e.g., $A(x)$ is "classically false" for some x, then we make sure that $A^d(x)$ is derviable as false for that same x from the rules, but not necessarily $A(x)$, thus allowing to detect potential MKNF-inconsistencies. That is also the reason why (i1) and (i2) do not produce the doubled counterparts: atoms based on predicates of the forms NC^d or NR^d are not used anywhere.

We can show that the translation also maintains derivability of atoms.

Lemma 14. *Let $\mathcal{O}$ be a reduced $\mathcal{EL}^+_\bot$ ontology, A a unary and R a binary predicate: $\mathcal{O} \models A(a)$ iff $\mathcal{P}^d_\mathcal{O} \models A(a)$ and $\mathcal{O}^d \models A^d(a)$ iff $\mathcal{P}^d_\mathcal{O} \models A^d(a)$, and, likewise, $\mathcal{O} \models R(a, b)$ iff $\mathcal{P}^d_\mathcal{O} \models R(a, b)$ and $\mathcal{O}^d \models R^d(a, b)$ iff $\mathcal{P}^d_\mathcal{O} \models R^d(a, b)$.*

Thus, we can define a correct partial oracle based on $\mathcal{P}^d_\mathcal{O}$.

Theorem 15. *Let $\mathcal{K}^d = (\mathcal{O}, \mathcal{O}^d, \mathcal{P}^d)$ be a doubled MKNF KB, $\mathcal{O}_1$ the reduced ontology of $\mathcal{O}$, and $pT^{\mathcal{EL}}_\mathcal{O}$ a* partial $\mathcal{EL}$ oracle *such that $pT^{\mathcal{EL}}_\mathcal{O}(\mathcal{I}, S, \mathcal{L})$ iff $\mathcal{P}^d_{\mathcal{O}_1} \cup \mathcal{I} \cup \mathcal{L} \models S$. Then $pT^{\mathcal{EL}}_\mathcal{O}$ is a correct partial oracle w.r.t. $compT_\mathcal{O}$.*[7]

Instead of coupling two rule reasoners that interact with each other using an oracle, we can simplify the process altogether and integrate both into one rule reasoner. The resulting approach is decidable with data complexity in $\mathbf{P}$.

Theorem 16. *Let $\mathcal{K} = (\mathcal{O}, \mathcal{P})$ be an MKNF KB with $\mathcal{EL}^+_\bot\mathcal{O}$. An* **SLG($\mathcal{O}$)** *evaluation of a query in $\mathcal{K}_{\mathcal{EL}^+_\bot} = (\emptyset, (\mathcal{P}^d \cup \mathcal{P}^d_\mathcal{O}))$ is decidable with data complexity in $\mathbf{P}$.*

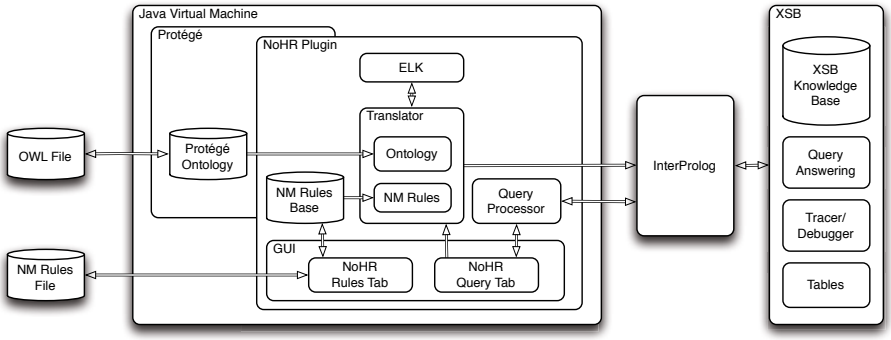


Fig. 1. System Architecture of NoHR

4 System Description

In this Section, we briefly describe the architecture of our plug-in for Protégé as shown in Fig. 1 and discuss some features of our implementation and querying in XSB.

The input for our plug-in consists of an OWL file, which can be manipulated as usual in Protégé, and a rule file. For the latter, we provide a tab called NoHR Rules that allows us to load, save and edit rule files in a text panel. The syntax follows Prolog conventions, so that one rule from Ex. 2 can be represented, e.g., by

```
SeaSideCity(X) :- PortCity(X), not NonSeaSideCity(X).
```

The NoHR Query tab also allows for the visualization of the rules, but its main purpose is to provide an interface for querying the combined KB. Whenever the first query is posed by pushing "Execute", the translator is started, initiating the ELK reasoner to classify the ontology and return the inferred axioms to translator. It is verified whether $DisjointWith$ axioms appear in $\mathcal{O}$ which determines whether the application of the transformations presented in Sect. 3 provides a doubled set of rules or not, and whether the non-monotonic rules have to be doubled as well. Then, accordingly, a joint (non-monotonic) rule set is created in which all predicates and constants are encoded using MD5 to ensure full compatibility with XSB Prolog's more restrictive admitted input syntax. The result is transfered to XSB via InterProlog [8], which is an open-source Java front-end allowing the communication between Java and a Prolog engine.

Next, the query is sent via InterProlog to XSB, and answers are returned to the query processor, which collects them and sets up a table showing for which variable substitutions we obtain true, undefined, or inconsistent valuations (or just shows the truth value for a ground query). The table itself is shown in the Result tab of the Output panel, while the Log tab shows measured times and system messages, including those from XSB via InterProlog. XSB itself not only answers queries very efficiently in a top-down manner, with tabling, it also avoids infinite loops.

Once the query has been answered, the user may pose other queries, and the system will simply send them directly without any repeated preprocessing. If the user changes

[7] Of course, the partial $\mathcal{EL}$ oracle has to be defined w.r.t. the reduced $\mathcal{O}_1$.

	# Axioms	ELK	Translator	XSB	Total
ChEBI	67184	3,92	7,41	1,75	9,16
EMAP	13730	2,94	5,14	0,00	5,14
Fly Anatomy	19211	2,70	4,99	0,83	5,82
FMA	126548	7,56	14,67	3,58	18,25
GALEN-OWL	36547	5,00	8,76	2,46	11,22
GALEN7	44461	5,88	9,67	4,56	14,23
GALEN8	73590	21,90	42,76	28,12	70,88
GO1	28897	3,74	6,06	1,09	7,15
GO2	73590	4,57	8,90	5,02	13,92
Molecule Role	9629	3,41	5,14	0,15	5,29
SNOMED CT	294480	20,61	43,32	24,69	68,01

Fig. 2. Preprocessing time (s) of different $\mathcal{EL}$ ontologies

data in the ontology or in the rules, then the system offers the option to recompile, but always restricted to the part that actually changed.

Our plug-in is under active development and the most recent version is available at `https://code.google.com/p/nohr-reasoner/`.

5 Evaluation

In this section, we evaluate our system with the aim of showing that a) different $\mathcal{EL}$ ontologies can be preprocessed for querying in a short period of time, b) adding rules increases the time of the translation only linearly, and c) querying time is in comparison to a) and b) in general completely neglectable.

We performed the tests on a Mac book air 13 under Mac OS X 10.8.4 with a 1.8 GHz Intel Core i5 processor and 8 GB 1600 MHz DDR3 of memory. We used OWL API 3.4.2 for managing ontologies, the ELK libraries 0.3.2, directly integrated into our tool, InterProlog 2.3a4 as Java front end between Java and Prolog, and XSB 3.4.0 for querying. We ran all tests in a terminal version and Java with the "-XX:+AggressiveHeap" option, and test results are averages over 5 runs.

First, we preprocessed a number of ontologies without additional rules and measured the time. The results are shown in Fig. 2.

We considered the ontologies, mentioned in [19], that are available online[8] or, in the case of SNOMED CT, freely available for research and evaluation.[9] The second column of Fig. 2 shows the number of axioms in each ontology to give an idea of their size. Very detailed measures of these ontologies are presented in [19].

We measured the time it takes to only classify these ontologies with the ELK standalone application for comparison (third column), and also the time to translate the

[8] `http://code.google.com/p/elk-reasoner/`
[9] `http://www.ihtsdo.org/licensing/`

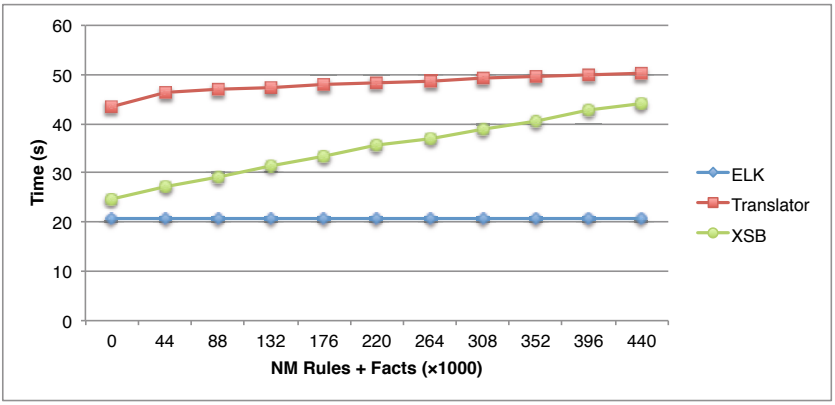


Fig. 3. Preprocessing time for SNOMED with a varying number of Rules

OWL file into a file that can be loaded into XSB (Translator, fourth column). Then, we measured the time to load the resulting file dynamically into XSB (fifth column) and we also show the total time (sixth column, sum of columns four and five).

One can observe that the total time is not proportional to the number of axioms, but rather also influenced by which kinds of axioms appear in each ontology which in particular affects the reasoning time of ELK (see also [19]). We can also see that the translator (including ELK processing) takes approximately twice the amount of time of ELK alone, while loading the file in XSB varies from 0 sec. (EMAP) to approximately the same time as running ELK alone, depending on the number of derived axioms that can be translated into rules. Still, the total preprocessing time varies from 5.14 sec. to 70.88 sec. which we belief is acceptable since this has to be executed only once before querying.

Next, we considered only SNOMED CT and added a varying number of non-monotonic rules. These rules were generated arbitrarily, using predicates from the ontology and additional new predicates (up to arity three), producing rules with a random number of body atoms varying from 1 to 10 and facts (rules without body atoms) with a ratio of 1:10. Note that, due to the translation of the DL part into rules, all atoms literally become non-DL-atoms. So ensuring that each variable appearing in the rule is contained in at least one non-negated body atom suffices to guarantee DL-safety for these rules.

The results are shown in Fig. 3 (containing also a constant line for classification of ELK alone and starting with the values from the first experiment with no additional rules), and clearly show that the time of translator and loading the file in XSB only grows linearly on the number of rules with a small degree, in particular in the case of translator. This indicates that adding non-monotonic rules to ontologies has a rather low impact on preprocessing.

Finally, we tested the querying time. To this purpose, we randomly generated and handcrafted several queries of different sizes and shapes, some satisfiable while others not, using SNOMED CT with a varying number of non-monotonic rules as described in

the previous experiment. More concretely, we tested queries ranging from single atoms to complex conjunctive queries. We also varied the depth of classes and properties in the hierarchies, how many sub-elements the considered classes/properties have themselves (the more they have, the more answers are returned w.r.t. the arbitrarily many created rules and facts), and how variables are connected. E.g., we considered sequences of atoms without any (positive variable) connection between them, thus creating all combinations of the answers for each atom, but also queries in which properties (or new predicates of arity greater 1) introduce connections or where the same variable appears in different queried classes. In all cases, we observed that the query response time is interactive, mostly significantly below one second, observing longer reply times only if the number of replies is very high because either the queried class contains many subclasses in the hierarchy or if the arbitrarily generated rules create too many meaningless links, thus in the worst case requiring to compute the entire model. Requesting only one solution avoids this problem. Additionally, the question of realistic randomly generated rule bodies for testing querying time remain an issue of future work.

6 Conclusions

We have presented NoHR, the first plug-in for the ontology editor Protégé that integrates non-monotonic rules and top-down queries with ontologies in the OWL 2 profile OWL 2 EL. Our approach realizes an **SLG**($\mathcal{O}$) oracle for $\mathcal{EL}^+_\bot$, utilizing the $\mathcal{EL}$ reasoner ELK for preprocessing an ontology and then translating it into rules, which can be queried together with the non-monotonic rules in XSB.

We have generalized the procedure presented in [2] to non-normalized $\mathcal{EL}^+_\bot$ knowledge bases and shown that this formalization provides a correct **SLG**($\mathcal{O}$) oracle. We also have discussed how this procedure is implemented in our tool, and that it offers the representation of non-monotonic knowledge such as defaults and exceptions in a seamless way. We have evaluated the performance showing that different $\mathcal{EL}$ ontologies can be preprocessed for querying in a short period of time, adding rules increases this time only linearly, and querying time is in comparison to preprocessing insignificant.

There are several relevant approaches discussed in the literature. Most closely related are probably [16,21], because both build on the well-founded MKNF semantics [20]. In fact, [16] is maybe closest in spirit to the original idea of **SLG**($\mathcal{O}$) oracles presented in [1]. It utilizes the CDF framework already integrated in XSB, but its non-standard language is a drawback if we want to achieve compatibility with standard OWL tools based on the OWL API. On the other hand, [21], presents an OWL 2 QL oracle based on common rewritings in the underlying DL DL-Lite [3]. Less closely related is the work pursued in [7,15] that investigates direct non-monotonic extensions of $\mathcal{EL}$, so that the main reasoning task focuses on finding default subset inclusions, unlike our query-centered approach.

Two related tools are DReW [26] and HD Rules [10], but both are based on different underlying formalisms to combine ontologies and non-monotonic rules. The former builds on dl-programs [12] and focuses on datalog-rewritable DLs [17], and the latter builds on Hybrid Rules [11]. While a more detailed comparison is surely of interest, the main problem is that both underlying formalisms differ from MKNF knowledge

bases in the way information can flow between its two components and how flexible the language is [12,24]. Finally, SWRL-IQ [13] is also interesting because it utilizes expressive features of XSB, but in a monotonic setting and under Protégé 3.X, so OWL 2 is not supported.

In terms of future work, we consider extending our tool so that it is possible to load files in RIF format, thereby achieving a tool to jointly utilize the two language paradigms present in the ongoing standardization of the Semantic Web. We also want to work on extending the language, to allow nominals, or at least safe nominals [19]. Finally, improving the user interface is also in our focus leveraging XSB language features, in particular in the light of [13], which uses an expressive query language and elaborate traces for finding explanations.

Acknowledgments. We would like to thank Miguel Calejo for his help with InterProlog, Pavel Klinov for his help with ELK, Terry Swift for his help with XSB, and Gonca Güllü for her collaboration. Vadim Ivanov was partially supported by a MULTIC – Erasmus Mundus Action 2 grant. Matthias Knorr and João Leite were partially supported by FCT funded project ERRO – Efficient Reasoning with Rules and Ontologies (PTDC/EIA-CCO/121823/2010) and Matthias Knorr also by FCT grant SFRH/BPD/86970/2012.

References

1. Alferes, J.J., Knorr, M., Swift, T.: Queries to hybrid MKNF knowledge bases through oracular tabling. In: Bernstein, A., Karger, D.R., Heath, T., Feigenbaum, L., Maynard, D., Motta, E., Thirunarayan, K. (eds.) ISWC 2009. LNCS, vol. 5823, pp. 1–16. Springer, Heidelberg (2009)
2. Alferes, J.J., Knorr, M., Swift, T.: Query-driven procedures for hybrid MKNF knowledge bases. ACM Trans. Comput. Log. 14(2), 1–43 (2013)
3. Artale, A., Calvanese, D., Kontchakov, R., Zakharyaschev, M.: The $DL\text{-}Lite$ family and relations. J. Artif. Intell. Res. (JAIR) 36, 1–69 (2009)
4. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F. (eds.): The Description Logic Handbook: Theory, Implementation, and Applications, 3rd edn. Cambridge University Press (2010)
5. Baader, F., Brandt, S., Lutz, C.: Pushing the el envelope. In: Kaelbling, L.P., Saffiotti, A. (eds.) IJCAI, pp. 364–369. Professional Book Center (2005)
6. Boley, H., Kifer, M. (eds.): RIF Overview. W3C Recommendation (February 5, 2013), `http://www.w3.org/TR/rif-overview/`
7. Bonatti, P.A., Faella, M., Sauro, L.: $\mathcal{EL}$ with default attributes and overriding. In: Patel-Schneider, P.F., Pan, Y., Hitzler, P., Mika, P., Zhang, L., Pan, J.Z., Horrocks, I., Glimm, B. (eds.) ISWC 2010, Part I. LNCS, vol. 6496, pp. 64–79. Springer, Heidelberg (2010)
8. Calejo, M.: Interprolog: Towards a declarative embedding of logic programming in java. In: Alferes, J.J., Leite, J. (eds.) JELIA 2004. LNCS (LNAI), vol. 3229, pp. 714–717. Springer, Heidelberg (2004)
9. Chen, W., Warren, D.S.: Tabled Evaluation with Delaying for General Logic Programs. J. ACM 43(1), 20–74 (1996)
10. Drabent, W., Henriksson, J., Maluszynski, J.: Hd-rules: A hybrid system interfacing prolog with dl-reasoners. In: Polleres, A., Pearce, D., Heymans, S., Ruckhaus, E. (eds.) ALPSWS. CEUR Workshop Proceedings, vol. 287, CEUR-WS.org (2007)

11. Drabent, W., Maluszynski, J.: Hybrid rules with well-founded semantics. Knowl. Inf. Syst. 25(1), 137–168 (2010)
12. Eiter, T., Ianni, G., Lukasiewicz, T., Schindlauer, R., Tompits, H.: Combining answer set programming with description logics for the semantic web. Artif. Intell. 172(12-13), 1495–1539 (2008)
13. Elenius, D.: Swrl-iq: A prolog-based query tool for owl and swrl. In: Klinov, P., Horridge, M. (eds.) OWLED. CEUR Workshop Proceedings, vol. 849, CEUR-WS.org (2012)
14. Gelder, A.V., Ross, K.A., Schlipf, J.S.: The well-founded semantics for general logic programs. J. ACM 38(3), 620–650 (1991)
15. Giordano, L., Gliozzi, V., Olivetti, N., Pozzato, G.L.: A tableau calculus for a nonmonotonic extension of $\mathcal{EL}^{\perp}$. In: Brünnler, K., Metcalfe, G. (eds.) TABLEAUX 2011. LNCS, vol. 6793, pp. 180–195. Springer, Heidelberg (2011)
16. Gomes, A.S., Alferes, J.J., Swift, T.: Implementing query answering for hybrid MKNF knowledge bases. In: Carro, M., Peña, R. (eds.) PADL 2010. LNCS, vol. 5937, pp. 25–39. Springer, Heidelberg (2010)
17. Heymans, S., Eiter, T., Xiao, G.: Tractable reasoning with dl-programs over datalog-rewritable description logics. In: Coelho, H., Studer, R., Wooldridge, M. (eds.) ECAI. Frontiers in Artificial Intelligence and Applications, vol. 215, pp. 35–40. IOS Press (2010)
18. Kazakov, Y., Krötzsch, M., Simančík, F.: Concurrent classification of $\mathcal{EL}$ ontologies. In: Aroyo, L., Welty, C., Alani, H., Taylor, J., Bernstein, A., Kagal, L., Noy, N., Blomqvist, E. (eds.) ISWC 2011, Part I. LNCS, vol. 7031, pp. 305–320. Springer, Heidelberg (2011)
19. Kazakov, Y., Krötzsch, M., Simančík, F.: The incredible elk. Tech. rep., University of Oxford, submitted to a journal (2013), `http://elk.semanticweb.org/papers/elk-journal-2013.pdf`
20. Knorr, M., Alferes, J.J., Hitzler, P.: Local closed world reasoning with description logics under the well-founded semantics. Artif. Intell. 175(9-10), 1528–1554 (2011)
21. Knorr, M., Alferes, J.J.: Querying owl 2 ql and non-monotonic rules. In: Aroyo, L., Welty, C., Alani, H., Taylor, J., Bernstein, A., Kagal, L., Noy, N., Blomqvist, E. (eds.) ISWC 2011, Part I. LNCS, vol. 7031, pp. 338–353. Springer, Heidelberg (2011)
22. Lifschitz, V.: Nonmonotonic databases and epistemic queries. In: Mylopoulos, J., Reiter, R. (eds.) IJCAI, pp. 381–386. Morgan Kaufmann (1991)
23. Motik, B., Cuenca Grau, B., Horrocks, I., Wu, Z., Fokoue, A., Lutz, C. (eds.): OWL 2 Web Ontology Language: Profiles. W3C Recommendation (February 5, 2013), `http://www.w3.org/TR/owl2-profiles/`
24. Motik, B., Rosati, R.: Reconciling description logics and rules. J. ACM 57(5) (2010)
25. Patel, C., Cimino, J.J., Dolby, J., Fokoue, A., Kalyanpur, A., Kershenbaum, A., Ma, L., Schonberg, E., Srinivas, K.: Matching patient records to clinical trials using ontologies. In: Aberer, K., Choi, K.-S., Noy, N., Allemang, D., Lee, K.-I., Nixon, L.J.B., Golbeck, J., Mika, P., Maynard, D., Mizoguchi, R., Schreiber, G., Cudré-Mauroux, P. (eds.) ASWC 2007 and ISWC 2007. LNCS, vol. 4825, pp. 816–829. Springer, Heidelberg (2007)
26. Xiao, G., Eiter, T., Heymans, S.: The DReW system for nonmonotonic dl-programs. In: SWWS 2012. Springer Proceedings in Complexity. Springer (2013)

Incremental Reasoning in OWL EL without Bookkeeping

Yevgeny Kazakov and Pavel Klinov

The University of Ulm, Germany
{yevgeny.kazakov,pavel.klinov}@uni-ulm.de

Abstract. We describe a method for updating the classification of ontologies expressed in the $\mathcal{EL}$ family of Description Logics after some axioms have been added or deleted. While incremental classification modulo additions is relatively straightforward, handling deletions is more problematic since it requires retracting logical consequences that are no longer valid. Known algorithms address this problem using various forms of bookkeeping to trace the consequences back to premises. But such additional data can consume memory and place an extra burden on the reasoner during application of inferences. In this paper, we present a technique, which avoids this extra cost while being very efficient for small incremental changes in ontologies. The technique is freely available as a part of the open-source $\mathcal{EL}$ reasoner ELK and its efficiency is demonstrated on naturally occurring and synthetic data.

1 Introduction and Motivation

The $\mathcal{EL}$ family of Description Logics (DLs) are tractable extensions of the DL $\mathcal{EL}$ featuring conjunction and existential restriction. Despite a limited number of constructors, $\mathcal{EL}$ became the language of choice in many applications, especially in Biology and Medicine, which require management of large terminologies. The DL $\mathcal{EL}^{++}$ [1]—an extension of $\mathcal{EL}$ with other features such as complex role inclusion axioms, nominals, and datatypes—became the basis of the OWL EL profile [2] of the Web ontology language OWL 2 specifically aimed at such applications.

Ontology classification is one of the main reasoning tasks. It requires computing all entailed (implicit) subsumption relations between atomic concepts. Specialized $\mathcal{EL}$ reasoners, such as CEL [3], ELK [4], jcel [5], and Snorocket [6] are able to compute the classification for ontologies as large as SNOMED CT [7] with about 300,000 axioms. Classification plays the key role during ontology development, e.g., for detecting modeling errors that result in mismatches between terms. But even with fast classification procedures, frequent re-classification of ontologies can introduce significant delays in the development workflow, especially as ontologies grow over time.

Several incremental reasoning procedures have been proposed to optimize frequent ontology re-classification after small changes. Most procedures maintain extra information to trace conclusions back to the axioms in order to deal with axiom deletions (see Section 2). Although managing this information typically incurs only a linear overhead, it can be a high cost for large ontologies such as SNOMED CT. In this paper, we propose an incremental reasoning method which does not require computing any of such

H. Alani et al. (Eds.): ISWC 2013, Part I, LNCS 8218, pp. 232–247, 2013.

information. The main idea is to split the derived conclusions into several partitions. We identify partitions containing 'affected' consequences (those that could be invalidated by deletion) using a simple forward chaining procedure, and then re-compute all conclusions in these partitions. This way, we avoid storing any bookkeeping information for checking whether the affected consequences still follow from other conclusions. Our hypothesis is that, if the number of partitions is sufficiently large, changes are relatively small, and most inferences happen within individual partitions, the re-computation of affected partitions will not be too expensive. We describe a particular partitioning method for $\mathcal{EL}$ that has this property, and verify our hypothesis experimentally. Our experiments demonstrate that for large ontologies, such as SNOMED CT, incremental classification can be 10–40 times faster than the (already highly optimized) full classification, thus making re-classification almost instantaneous.

In this paper we focus on the DL $\mathcal{EL}^+$, which covers most of the existing OWL EL ontologies, and for simplicity, consider only additions and deletions of concept axioms, but not of role axioms. Although the method can be extended to changes in role axioms, it is unlikely to pay off in practice, because such changes are more likely to cause a significant impact on the result of the classification.

2 Related Work

Directly relevant to this work are various extensions to DL reasoning algorithms to support incremental changes.

Incremental classification in $\mathcal{EL}$ modulo additions implemented in the CEL system, comes closest [8]. The procedure works, essentially, by applying new inferences corresponding to the added axioms and closing the set of old and new conclusions under all inference rules. Deletion of axioms is not supported.

Known algorithms that support deletions require a form of bookkeeping to trace conclusions back to the premises. The Pellet reasoner [9] implements a technique called *tableau tracing* to keep track of the axioms used in tableau inferences [10]. Tracing maps tableau elements (nodes, labels, and relations) to the responsible axioms. Upon deletion of axioms, the corresponding elements get deleted. This method is memory-intensive for large tableaux and currently supports only ABox changes.

The *module-based* incremental reasoning method does not perform full tracing of inferences, but instead maintains a collection of modules for derived conclusions [11]. The modules consist of axioms in the ontology that entail the respective conclusion, but they are not necessarily minimal. If no axiom in the module was deleted then the entailment is still valid. Unlike tracing, the method does not require changes to the reasoning algorithm, but still incurs the cost of computing and storing the modules.

The approach presented in this paper is closely related to the classical DRed (*over-delete, re-derive*) strategy for incremental maintenance of recursive views in databases [12]. In the context of ontologies, this method was applied, e.g., for incremental updates of assertions materialized using datalog rules [13], and for *stream reasoning* in RDF [14]. Just like in DRed, we over-delete conclusions that were derived using deleted inferences (to be on the safe side), but instead of checking which deleted conclusions are still derivable using remaining inferences (which would require additional bookkeeping information), we re-compute some well-defined subset of 'broken' conclusions.

Table 1. The syntax and semantics of $\mathcal{EL}^+$

	Syntax	Semantics
Roles:		
atomic role	R	$R^{\mathcal{I}}$
Concepts:		
atomic concept	A	$A^{\mathcal{I}}$
top	$\top$	$\Delta^{\mathcal{I}}$
bottom	$\bot$	$\emptyset$
conjunction	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
existential restriction	$\exists R.C$	$\{x \mid \exists y \in C^{\mathcal{I}} : \langle x, y\rangle \in R^{\mathcal{I}}\}$
Axioms:		
concept inclusion	$C \sqsubseteq D$	$C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
role inclusion	$R \sqsubseteq S$	$R^{\mathcal{I}} \subseteq S^{\mathcal{I}}$
role composition	$R_1 \circ R_2 \sqsubseteq S$	$R_1^{\mathcal{I}} \circ R_2^{\mathcal{I}} \subseteq S^{\mathcal{I}}$

3 Preliminaries

3.1 The Description Logic $\mathcal{EL}^+$

In this paper, we will focus on the DL $\mathcal{EL}^+$ [3], which can be seen as $\mathcal{EL}^{++}$ [1] without nominals, datatypes, and the bottom concept $\bot$. It is defined w.r.t. a vocabulary consisting of countably infinite sets of *(atomic) roles* and *atomic concepts*. Complex *concepts* and *axioms* are defined recursively in Table 1. We use the letters R, S for roles, C, D, E for concepts, and A, B for atomic concepts. An *ontology* is a finite set of axioms. Given an ontology $\mathcal{O}$, we write $\sqsubseteq^*_{\mathcal{O}}$ for the smallest reflexive transitive binary relation over roles such that $R \sqsubseteq^*_{\mathcal{O}} S$ holds for all $R \sqsubseteq S \in \mathcal{O}$.

An *interpretation* $\mathcal{I}$ consists of a nonempty set $\Delta^{\mathcal{I}}$ called the *domain* of $\mathcal{I}$ and an interpretation function $\cdot^{\mathcal{I}}$ that assigns to each role R a binary relation $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$, and to each atomic concept A a set $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$. This assignment is extended to complex concepts as shown in Table 1. $\mathcal{I}$ *satisfies* an axiom α (written $\mathcal{I} \models \alpha$) if the corresponding condition in Table 1 holds. $\mathcal{I}$ is a *model* of an ontology $\mathcal{O}$ (written $\mathcal{I} \models \mathcal{O}$) if $\mathcal{I}$ satisfies all axioms in $\mathcal{O}$. We say that $\mathcal{O}$ *entails* an axiom α (written $\mathcal{O} \models \alpha$), if every model of $\mathcal{O}$ satisfies α. The *ontology classification task* requires to compute all entailed subsumptions between atomic concepts occurring in $\mathcal{O}$.

3.2 Inferences and Inference Rules

Let **Exp** be a fixed countable set of *expressions*. An *inference* over **Exp** is an object *inf* which is assigned with a finite set of *premises inf*.Premises $\subseteq$ **Exp** and a *conclusion inf*.conclusion $\in$ **Exp**. When *inf*.Premises $= \emptyset$, we say that *inf* is an *initialization inference*. An *inference rule* R over **Exp** is a countable set of inferences over **Exp**; it is an *initialization rule* if all these inferences are initialization inferences. The *cardinality of the rule* R (notation $\|\mathsf{R}\|$) is the number of inferences *inf* $\in \mathsf{R}$. In this paper, we view an *inference system* as one inference rule R representing all of their inferences.

$$\mathbf{R_0}\ \frac{}{C \sqsubseteq C}: C \text{ occurs in } \mathcal{O} \qquad \mathbf{R}^+_\sqcap\ \frac{C \sqsubseteq D_1 \quad C \sqsubseteq D_2}{C \sqsubseteq D_1 \sqcap D_2}: D_1 \sqcap D_2 \text{ occurs in } \mathcal{O}$$

$$\mathbf{R}_\top\ \frac{}{C \sqsubseteq \top}: C \text{ and } \top \text{ occur in } \mathcal{O} \qquad \mathbf{R}_\exists\ \frac{E \sqsubseteq \exists R.C \quad C \sqsubseteq D}{E \sqsubseteq \exists S.D}: \begin{array}{l}\exists S.D \text{ occurs in } \mathcal{O}\\ R \sqsubseteq^*_{\mathcal{O}} S\end{array}$$

$$\mathbf{R}_\sqsubseteq\ \frac{C \sqsubseteq D}{C \sqsubseteq E}: D \sqsubseteq E \in \mathcal{O} \qquad \mathbf{R}_\circ\ \frac{E \sqsubseteq \exists R_1.C \quad C \sqsubseteq \exists R_2.D}{E \sqsubseteq \exists S.D}: \begin{array}{l}S_1 \circ S_2 \sqsubseteq S \in \mathcal{O}\\ R_1 \sqsubseteq^*_{\mathcal{O}} S_1\\ R_2 \sqsubseteq^*_{\mathcal{O}} S_2\end{array}$$

$$\mathbf{R}^-_\sqcap\ \frac{C \sqsubseteq D_1 \sqcap D_2}{C \sqsubseteq D_1 \quad C \sqsubseteq D_2}$$

Fig. 1. The inference rules for reasoning in $\mathcal{EL}^+$

We say that a set of expressions $Exp \subseteq \mathbf{Exp}$ is *closed under an inference inf* if *inf*.Premises $\subseteq Exp$ implies *inf*.conclusion $\in Exp$. Exp is *closed under an inference rule* R if Exp is closed under every inference $inf \in \mathsf{R}$. *The closure under* R is the smallest set of expressions closed under R. Note that the closure is always empty if R does not contain initialization inferences.

We will often restrict inference rules to subsets of premises. Let $Exp \subseteq \mathbf{Exp}$ be a set of expressions, and R an inference rule. By $\mathsf{R}(Exp)$ ($\mathsf{R}[Exp]$) we denote the rule consisting of all inferences $inf \in \mathsf{R}$ such that *inf*.Premises $\subseteq Exp$ (respectively $Exp \subseteq$ *inf*.Premises). We can combine these operators: for example, $\mathsf{R}[Exp_1](Exp_2)$ consists of those inferences in R whose premises contain all expressions from Exp_1 and are a subset of Exp_2. Note that this is the same as $\mathsf{R}(Exp_2)[Exp_1]$. For simplicity, we write $\mathsf{R}()$, $\mathsf{R}[]$, $\mathsf{R}(exp)$, and $\mathsf{R}[exp]$ instead of $\mathsf{R}(\emptyset)$, $\mathsf{R}[\emptyset]$, $\mathsf{R}(\{exp\})$, and $\mathsf{R}[\{exp\}]$ respectively. Note that $\mathsf{R}[] = \mathsf{R}$ and $\mathsf{R}()$ consists of all initialization inferences in R.

3.3 The Reasoning Procedure for $\mathcal{EL}^+$

The $\mathcal{EL}^+$ reasoning procedure works by applying inference rules to derive subsumptions between concepts. In this paper, we use the rules from $\mathcal{EL}^{++}$ [1] restricted to $\mathcal{EL}^+$, but present them in a way that does not require the normalization stage [4].

The rules for $\mathcal{EL}^+$ are given in Figure 1, where the premises (if any) are given above the horizontal line, and the conclusions below. Some rules have side conditions given after the colon that restrict the expressions to which the rules are applicable. For example, rule $\mathbf{R}^+_\sqcap$ contains one inference *inf* for each C, D_1, D_2, such that $D_1 \sqcap D_2$ occurs in $\mathcal{O}$ with *inf*.Premises $= \{C \sqsubseteq D_1,\ C \sqsubseteq D_2\}$, *inf*.conclusion $= C \sqsubseteq D_1 \sqcap D_2$. Note that the axioms in the ontology $\mathcal{O}$ are only used in side conditions of the rules and never used as premises of the rules.

The rules in Figure 1 are complete for deriving subsumptions between the concepts occurring in the ontology. That is, if $\mathcal{O} \models C \sqsubseteq D$ for C and D occurring in $\mathcal{O}$, then $C \sqsubseteq D$ can be derived using the rules in Figure 1 [1]. Therefore, in order to classify the ontology, it is sufficient to compute the closure under the rules and take the derived subsumptions between atomic concepts. The following example illustrates the application of rules in Figure 1 for deriving the entailed subsumption relations.

Example 1. Consider the following $\mathcal{EL}^+$ ontology $\mathcal{O}$:
(ax1): $A \sqsubseteq \exists R.B$ (ax2): $\exists H.B \sqsubseteq C$ (ax3): $R \sqsubseteq H$
(ax4): $B \sqsubseteq \exists S.A$ (ax5): $\exists S.C \sqsubseteq C$
The subsumptions below can be derived via rules in Figure 1:

$$A \sqsubseteq A \quad \text{by } \mathbf{R_0} \text{ since } A \text{ occurs in } \mathcal{O}, \quad (1)$$
$$B \sqsubseteq B \quad \text{by } \mathbf{R_0} \text{ since } B \text{ occurs in } \mathcal{O}, \quad (2)$$
$$C \sqsubseteq C \quad \text{by } \mathbf{R_0} \text{ since } C \text{ occurs in } \mathcal{O}, \quad (3)$$
$$\exists R.B \sqsubseteq \exists R.B \quad \text{by } \mathbf{R_0} \text{ since } \exists R.B \text{ occurs in } \mathcal{O}, \quad (4)$$
$$\exists S.A \sqsubseteq \exists S.A \quad \text{by } \mathbf{R_0} \text{ since } \exists S.A \text{ occurs in } \mathcal{O}, \quad (5)$$
$$\exists H.B \sqsubseteq \exists H.B \quad \text{by } \mathbf{R_0} \text{ since } \exists H.B \text{ occurs in } \mathcal{O}, \quad (6)$$
$$\exists S.C \sqsubseteq \exists S.C \quad \text{by } \mathbf{R_0} \text{ since } \exists S.C \text{ occurs in } \mathcal{O}, \quad (7)$$
$$A \sqsubseteq \exists R.B \quad \text{by } \mathbf{R}_\sqsubseteq \text{ to (1) using (ax1)}, \quad (8)$$
$$B \sqsubseteq \exists S.A \quad \text{by } \mathbf{R}_\sqsubseteq \text{ to (2) using (ax4)}, \quad (9)$$
$$\exists R.B \sqsubseteq \exists H.B \quad \text{by } \mathbf{R}_\exists \text{ to (4) and (2) using (ax3)}, \quad (10)$$
$$\exists H.B \sqsubseteq C \quad \text{by } \mathbf{R}_\sqsubseteq \text{ to (6) using (ax2)}, \quad (11)$$
$$\exists S.C \sqsubseteq C \quad \text{by } \mathbf{R}_\sqsubseteq \text{ to (7) using (ax5)}, \quad (12)$$
$$A \sqsubseteq \exists H.B \quad \text{by } \mathbf{R}_\exists \text{ to (8) and (2) using (ax3)}, \quad (13)$$
$$\exists R.B \sqsubseteq C \quad \text{by } \mathbf{R}_\sqsubseteq \text{ to (10) using (ax2)}, \quad (14)$$
$$A \sqsubseteq C \quad \text{by } \mathbf{R}_\sqsubseteq \text{ to (13) using (ax2)}, \quad (15)$$
$$\exists S.A \sqsubseteq \exists S.C \quad \text{by } \mathbf{R}_\exists \text{ to (5) and (15)}, \quad (16)$$
$$B \sqsubseteq \exists S.C \quad \text{by } \mathbf{R}_\exists \text{ to (9) and (15)}, \quad (17)$$
$$\exists S.A \sqsubseteq C \quad \text{by } \mathbf{R}_\sqsubseteq \text{ to (16) using (ax5)}, \quad (18)$$
$$B \sqsubseteq C \quad \text{by } \mathbf{R}_\sqsubseteq \text{ to (17) using (ax5)}. \quad (19)$$

The subsumptions (1)–(19) are closed under these rules so, by completeness, $A \sqsubseteq A$, $B \sqsubseteq B$, $C \sqsubseteq C$, $A \sqsubseteq C$, $B \sqsubseteq C$ are all atomic subsumptions entailed by $\mathcal{O}$.

3.4 Computing the Closure under Inference Rules

Computing the closure under inference rules, such as in Figure 1, can be performed using a well-known *forward chaining procedure* presented in Algorithm 1. The algorithm derives consequences by applying inferences in R and collects those conclusions between which all inferences are applied in a set Closure and the remaining ones in a queue Todo. The algorithm first initializes Todo with conclusions of the initialization inferences $\mathsf{R}() \subseteq \mathsf{R}$ (lines 2–3), and then in a cycle (lines 4–9), repeatedly takes the next expression $exp \in$ Todo, if any, inserts it into Closure if it does not occur there, and applies all inferences $inf \in \mathsf{R}[exp](\mathsf{Closure})$ having this expression as one of the premises and other premises from Closure. The conclusions of such inferences are then inserted back into Todo.

Algorithm 1. Computing the inference closure

```
   input  : R: a set of inferences
   output : Closure: the closure under R
1  Closure, Todo ← ∅;
2  for inf ∈ R() do                                   /* initialize */
3      Todo.add(inf.conclusion);
4  while Todo ≠ ∅ do                                  /* close */
5      exp ← Todo.takeNext();
6      if exp ∉ Closure then
7          Closure.add(exp);
8          for inf ∈ R[exp](Closure) do
9              Todo.add(inf.conclusion);
10 return Closure;
```

The following example illustrates the execution of Algorithm 1 for computing the deductive closure under inferences in Figure 1.

Example 2 (Example 1 continued). The conclusions (1)–(19) in Example 1 are already listed in the order in which they would be inserted into Todo by Algorithm 1. When a conclusion is inserted into Closure, all inferences involving this and the previous conclusions are applied. For example, when (10) is inserted, the previous conclusions (1)–(9) are already in Closure, so (14) is derived and added into Todo after (11)–(13).

Note that Algorithm 1 performs as many insertions into Todo as there are inferences in $\mathsf{R}(\mathsf{Closure}')$ for the result $\mathsf{Closure}'$ because every inference $inf \in \mathsf{R}(\mathsf{Closure}')$ is eventually applied, and an inference cannot apply more than once. Therefore, the number of inferences performed by Algorithm 1 is exactly $\|\mathsf{R}(\mathsf{Closure}')\|$. The time complexity of the algorithm depends highly on the representation of the inference rules. If the initialization inferences $inf \in \mathsf{R}()$ in line 2 and matching inferences $inf \in \mathsf{R}[exp](\mathsf{Closure})$ in line 8 can be effectively enumerated, the algorithm runs in $O(\|\mathsf{R}(\mathsf{Closure}')\|)$.

4 Incremental Deductive Closure Computation

In this section, we discuss algorithms for updating the deductive closure under a set of inferences after the set of inferences has changed. Just like in Section 3.4, the material in this section is not specific to any particular inference system, i.e., does not rely on the $\mathcal{EL}^+$ classification procedure described in Section 3.3.

The problem of incremental computation of the deductive closure can be formulated as follows. Let R, R^+ and R^- be sets of inferences, and Closure the closure under R. The objective is to compute the closure under the inferences in $(\mathsf{R} \setminus \mathsf{R}^-) \cup \mathsf{R}^+$, using Closure, R, R^+, or R^-, if necessary.

Algorithm 2. Update modulo additions

input : R, R^+: sets of inferences, Closure: the closure under R
output : Closure: the closure under $\mathsf{R} \cup \mathsf{R}^+$

```
1  Todo ← ∅;
2  for inf ∈ (R⁺ \ R)(Closure) do                          /* initialize */
3      Todo.add(inf.conclusion);
4  R ← R ∪ R⁺;
5  while Todo ≠ ∅ do                                        /* close */
6      exp ← Todo.takeNext();
7      if exp ∉ Closure then
8          Closure.add(exp);
9          for inf ∈ R[exp](Closure) do
10             Todo.add(inf.conclusion);
11 return Closure;
```

4.1 Additions Are Easy

If there are no deletions ($\mathsf{R}^- = \emptyset$), the closure under $\mathsf{R} \cup \mathsf{R}^+$ can be computed by Algorithm 2. Starting from Closure, the closure under R, the algorithm first initializes Todo with conclusions of new inferences $\mathit{inf} \in (\mathsf{R}^+ \setminus \mathsf{R})$ applicable to Closure, and then processes this queue with respect to the union of all inferences $\mathsf{R} \cup \mathsf{R}^+$ as it is done in Algorithm 1. Note that Algorithm 1 is just a special case of Algorithm 2 when Closure $= \emptyset$ and the initial set of inferences R is empty.

Let Closure be the set in the input of Algorithm 2, and Closure$'$ the set obtained in the output. Intuitively, the algorithm applies all inferences in $(\mathsf{R} \cup \mathsf{R}^+)(\mathsf{Closure}')$ that are not in $\mathsf{R}(\mathsf{Closure})$ because those should have been already applied. If, in contrast, we compute the closure from scratch using Algorithm 1, we would need to apply all inferences in $(\mathsf{R} \cup \mathsf{R}^+)(\mathsf{Closure}')$. Note that it is essential that Algorithm 2 starts with the closure under R. If we start with a set that is not closed under R, we may lose some conclusions because no inference in $\mathsf{R}(\mathsf{Closure})$ is applied by the algorithm.

4.2 Deletions Are Difficult

Let us now see how to update the closure under deletions, i.e., when $\mathsf{R}^+ = \emptyset$. Consider Algorithm 3, which works analogously to Algorithm 2, but removes conclusions instead of adding them. In this algorithm, the queue Todo is used to buffer conclusions that should be removed from Closure. We first initialize Todo with consequences of the removed inferences $\mathit{inf} \in \mathsf{R}^-(\mathsf{Closure})$ (lines 2–3), and then remove such elements from Closure together with the conclusions of inferences from Closure in which they participate (lines 5–10). Note that in this loop, it is sufficient to consider only consequences under the resulting $\mathsf{R} = \mathsf{R} \setminus \mathsf{R}^-$ because all consequences under R^- are already added into Todo during the initialization stage (lines 2–3).

Unfortunately, Algorithm 3 might not produce the closure under $\mathsf{R} \setminus \mathsf{R}^-$: it may delete expressions that are still derivable in $\mathsf{R} \setminus \mathsf{R}^-$. For example, for the input

Algorithm 3. Update modulo deletions (incomplete)

input : R, R^-: sets of inferences, $\mathsf{Closure}$: the closure under R
output : $\mathsf{Closure}$: a subset of the closure under $(\mathsf{R} \setminus \mathsf{R}^-)(\mathsf{Closure})$

```
1  Todo ← ∅;
2  for inf ∈ R⁻(Closure) do                                  /* initialize */
3      Todo.add(inf.conclusion);
4  R ← R \ R⁻;
5  while Todo ≠ ∅ do                                          /* close */
6      exp ← Todo.takeNext();
7      if exp ∈ Closure then
8          for inf ∈ R[exp](Closure) do
9              Todo.add(inf.conclusion);
10         Closure.remove(exp);
11 return Closure;
```

$\mathsf{R} = \{/a,\ a/b,\ b/a\}$ (x/y is an inference with the premise x and conclusion y), $\mathsf{R}^- = \{b/a\}$, and $\mathsf{Closure} = \{a,\ b\}$, Algorithm 3 removes both a since it is a conclusion of $\mathsf{R}^-(\mathsf{Closure})$, and b since it is a conclusion of $(\mathsf{R} \setminus \mathsf{R}^-)[a](\mathsf{Closure})$, yet both a and b are still derivable by the remaining inferences $\mathsf{R} \setminus \mathsf{R}^- = \{/a,\ a/b\}$.

A common solution to this problem is to check which of the removed expressions are conclusions of the remaining inferences in $\mathsf{R}(\mathsf{Closure})$, put them back into Todo, and re-apply the inferences for them like in the main loop of Algorithm 2 (lines 5–10). This is known as the DRed (over-delete, re-derive) strategy in logic programming [12]. To check whether an expression is a conclusion of some inference from $\mathsf{Closure}$, however, one either needs to record how conclusions where produced, or build indexes that help to identify matching premises in $\mathsf{Closure}$ by conclusions. Storing this information for everything derived can consume a lot of memory and slow down the inference process.

Note that it makes little sense to 'simply re-apply' all inferences in R to the set $\mathsf{Closure}$ produced by Algorithm 3. This differs little from running Algorithm 1 from scratch, which applies exactly the same inferences anyway. Most of the inferences are likely to be already applied to $\mathsf{Closure}$, so, even if it is not 'fully' closed under R, it may be 'almost' closed. The main idea behind our method presented in the next section, is to identify a large enough subset of expressions $\mathsf{Closure}_1 \subseteq \mathsf{Closure}$ and a large enough subset of inferences $\mathsf{R}_1 \subseteq \mathsf{R}$, such that $\mathsf{Closure}_1$ is already closed under R_1. We can then re-compute the closure under R incrementally from $\mathsf{Closure}_1$ using Algorithm 2 for $\mathsf{R}^+ = \mathsf{R} \setminus \mathsf{R}_1$. As has been shown, using this approach we can avoid applying the already applied inferences in $\mathsf{R}_1(\mathsf{Closure}_1)$.

Let $\mathsf{Closure}$ be the set in the input of Algorithm 3, and $\mathsf{Closure}'$ the set obtained in the output. Similarly to Algorithm 2, Algorithm 3 applies all inferences in $\mathsf{R}(\mathsf{Closure})$ except for those in $(\mathsf{R} \setminus \mathsf{R}^-)(\mathsf{Closure}')$. Indeed, during initialization (lines 2–3) the algorithm applies all inferences in $\mathsf{R}^-(\mathsf{Closure})$, and in the main loop (lines 5–10) it applies each inference in $(\mathsf{R} \setminus \mathsf{R}^-)(\mathsf{Closure})$ that is not in $(\mathsf{R} \setminus \mathsf{R}^-)(\mathsf{Closure}')$—exactly those inferences that have at least one premise in $\mathsf{Closure} \setminus \mathsf{Closure}'$. The conclusion of

every such inference is removed from Closure, i.e., it is an element of $\mathsf{Closure} \setminus \mathsf{Closure}'$. Although, as has been pointed out, the output $\mathsf{Closure}'$ is not necessarily the closure under $\mathsf{R} \setminus \mathsf{R}^-$, it is, nevertheless, a subset of this closure:

Lemma 1. *Let* Closure *be the set in the input of Algorithm 3, and* $\mathsf{Closure}'$ *the set obtained in the output. Then* $\mathsf{Closure}'$ *is a subset of the closure under* $(\mathsf{R} \setminus \mathsf{R}^-)(\mathsf{Closure}')$.

Proof. Let $\mathsf{Closure}''$ be the closure under $(\mathsf{R} \setminus \mathsf{R}^-)(\mathsf{Closure}')$. We need to prove that $\mathsf{Closure}' \subseteq \mathsf{Closure}''$. Clearly, $\mathsf{Closure}' \subseteq \mathsf{Closure}$ and $\mathsf{Closure}'' \subseteq \mathsf{Closure}$. Define $\mathsf{Closure}_1 := (\mathsf{Closure} \setminus \mathsf{Closure}') \cup \mathsf{Closure}'' \subseteq \mathsf{Closure}$. We claim that $\mathsf{Closure}_1$ is closed under R. Indeed, take any $\mathit{inf} \in \mathsf{R}(\mathsf{Closure}_1)$. Then there are two cases possible:

1. $\mathit{inf} \in \mathsf{R}(\mathsf{Closure}) \setminus (\mathsf{R} \setminus \mathsf{R}^-)(\mathsf{Closure}')$: Then *inf* was applied in Algorithm 3. Therefore, *inf*.conclusion $\in \mathsf{Closure} \setminus \mathsf{Closure}' \subseteq \mathsf{Closure}_1$.
2. $\mathit{inf} \in (\mathsf{R} \setminus \mathsf{R}^-)(\mathsf{Closure}')$: Since $\mathit{inf} \in \mathsf{R}(\mathsf{Closure}_1)$ and $\mathsf{Closure}' \cap \mathsf{Closure}_1 \subseteq \mathsf{Closure}''$, we have $\mathit{inf} \in (\mathsf{R} \setminus \mathsf{R}^-)(\mathsf{Closure}'')$. Since $\mathsf{Closure}'' \subseteq \mathsf{Closure}_1$ and $\mathsf{Closure}''$ is closed under $(\mathsf{R} \setminus \mathsf{R}^-)(\mathsf{Closure}_1)$, then $\mathsf{Closure}''$ is closed under *inf*. Therefore *inf*.conclusion $\in \mathsf{Closure}'' \subseteq \mathsf{Closure}_1$.

Now, since $\mathsf{Closure}_1 \subseteq \mathsf{Closure}$ is closed under R and Closure is the smallest set closed under R, we have $\mathsf{Closure}_1 = \mathsf{Closure}$. Therefore, $\emptyset = \mathsf{Closure} \setminus \mathsf{Closure}_1 = \mathsf{Closure}' \setminus \mathsf{Closure}''$, and so, $\mathsf{Closure}' \subseteq \mathsf{Closure}''$, as required. □

Note that Lemma 1 claims something stronger than just that $\mathsf{Closure}'$ is a subset of the closure under $\mathsf{R} \setminus \mathsf{R}^-$. It is, in fact, a subset of the closure under $(\mathsf{R} \setminus \mathsf{R}^-)(\mathsf{Closure}') \subseteq \mathsf{R} \setminus \mathsf{R}^-$. Not every subset of the closure under $\mathsf{R} \setminus \mathsf{R}^-$ has this property. Intuitively, this property means that every expression in $\mathsf{Closure}'$ can be derived by inferences in $\mathsf{R} \setminus \mathsf{R}^-$ using only expressions in $\mathsf{Closure}'$ as intermediate conclusions. This property will be important for correctness of our method.

4.3 Incremental Updates Using Partitions

Our new method for updating the closure under deletions can be described as follows. We partition the set of expressions in Closure on disjoint subsets and modify Algorithm 3 such that whenever an expression is removed from Closure, its partition is marked as 'broken'. We then re-apply inferences that can produce conclusions in broken partitions to 'repair' the closure.

Formally, let **Pts** be a fixed countable set of *partition identifiers* (short *partitions*), and every expression $\mathit{exp} \in \mathbf{Exp}$ be assigned with exactly one partition exp.partition $\in$ **Pts**. For an inference rule R and a set of partitions $\mathit{Pts} \subseteq \mathbf{Pts}$, let $\mathsf{R}\langle \mathit{Pts} \rangle$ be the set of inferences $\mathit{inf} \in \mathsf{R}$ such that *inf*.conclusion.partition $\in \mathit{Pts}$ and exp.partition $\notin \mathit{Pts}$ for every $\mathit{exp} \in \mathit{inf}$.Premises. Intuitively, these are all inferences in R that can derive an expression whose partition is in *Pts* from expressions whose partitions are not in *Pts*.

We modify Algorithm 3 such that whenever an expression *exp* is removed from Closure in line 10, we add *exp*.partition into a special set of partitions Broken. This set is then used to repair Closure in Algorithm 4. The goal of the algorithm is to collect in the queue Todo the conclusions of inferences in $\mathsf{R}(\mathsf{Closure})$ that are missing in Closure.

Algorithm 4. Repair of over-deletions

input : R: a set of inferences, Closure: a subset of the closure under R (Closure), Broken: a set of partitions such that if *inf* ∈ R(Closure) and *inf*.conclusion ∉ Closure then *inf*.conclusion.partition ∈ Broken

output : Todo: the conclusions of inferences in R (Closure) that do not occur in Closure

```
1  Todo, ToRepair, Repaired ← ∅;
2  for inf ∈ R⟨Broken⟩(Closure) do                          /* initialize */
3      if inf.conclusion ∉ Closure then
4          Todo.add(inf.conclusion);
5      else
6          ToRepair.add(inf.conclusion);
7  while ToRepair ≠ ∅ do                                    /* close */
8      exp ← ToRepair.takeNext();
9      if exp ∉ Repaired then
10         for inf ∈ R[exp](Closure) do
11             if inf.conclusion.partition ∈ Broken then
12                 if inf.conclusion ∉ Closure then
13                     Todo.add(inf.conclusion);
14                 else
15                     ToRepair.add(inf.conclusion);
16         Repaired.add(exp);
17 return Todo;
```

This is done by applying all possible inferences $inf \in \mathsf{R}(\mathsf{Closure})$ that can produce such conclusions. There can be two types of such inferences: those whose premises do not belong to any partition in Broken, and those that have at least one such premise. The inferences of the first type are $\mathsf{R}\langle\mathsf{Broken}\rangle(\mathsf{Closure})$; they are applied in initialization (lines 2–6). The inferences of the second type are applied in the main loop of the algorithm (lines 7–16) to the respective expression in Closure whose partition is in Broken.

Whenever an inference *inf* is applied and *inf*.conclusion belongs to a partition in Broken (note that it is always the case for $inf \in \mathsf{R}\langle\mathsf{Broken}\rangle(\mathsf{Closure})$, see also line 11), we check if *inf*.conclusion occurs in Closure or not. If it does not occur, then we put the conclusion into the output Todo (lines 4, 13). Otherwise, we put it into a special queue ToRepair (lines 6, 15), and repeatedly apply for each $exp \in \mathsf{ToRepair}$ all inferences $inf \in \mathsf{R}[exp](\mathsf{Closure})$ of the second type in the main loop of the algorithm (lines 7–16). After applying all inferences, we move *exp* into a special set Repaired (line 16), which is there to make sure that we never consider *exp* again (see line 9).

Lemma 2. *Let* R, Closure, *and* Broken *be the inputs of Algorithm 4, and* Todo *the output. Then* $\mathsf{Todo} = \{inf.\mathsf{conclusion} \mid inf \in \mathsf{R}(\mathsf{Closure})\} \setminus \mathsf{Closure}$.

Proof. Let $\mathsf{Closure}' = \{inf.\mathsf{conclusion} \mid inf \in \mathsf{R}(\mathsf{Closure})\}$. We need to demonstrate that $\mathsf{Todo} = \mathsf{Closure}' \setminus \mathsf{Closure}$. Since $\mathsf{Todo} \subseteq \mathsf{Closure}'$ and $\mathsf{Closure} \cap \mathsf{Todo} = \emptyset$, it is sufficient to prove that $\mathsf{Closure}' \setminus \mathsf{Closure} \subseteq \mathsf{Todo}$.

First, note that $\mathsf{Closure}'$ is the closure under $\mathsf{R}(\mathsf{Closure})$. Indeed, if $\mathsf{Closure}''$ is the closure under $\mathsf{R}(\mathsf{Closure})$, then $\mathsf{Closure} \subseteq \mathsf{Closure}''$ by the assumption of Algorithm 4. Hence, for every $\mathit{inf} \in \mathsf{R}(\mathsf{Closure}) \subseteq \mathsf{R}(\mathsf{Closure}'')$, we have $\mathit{inf}.\mathsf{conclusion} \in \mathsf{Closure}''$. Therefore, $\mathsf{Closure}' \subseteq \mathsf{Closure}''$, and since $\mathsf{Closure}'$ is closed under $\mathsf{R}(\mathsf{Closure})$, we have $\mathsf{Closure}' = \mathsf{Closure}''$.

Let $\mathsf{Closure}_1 = \{\mathit{exp} \in \mathsf{Closure} \mid \mathit{exp}.\mathsf{partition} \notin \mathsf{Broken}\}$. Then it is easy to see from Algorithm 4 that for every $\mathit{inf} \in \mathsf{R}(\mathsf{Closure}_1 \cup \mathsf{Repaired})$, we have $\mathit{inf}.\mathsf{conclusion} \in \mathsf{Closure}_1 \cup \mathsf{Repaired} \cup \mathsf{Todo}$. Indeed, if $\mathit{inf}.\mathsf{conclusion}.\mathsf{partition} \notin \mathsf{Broken}$ then by assumption of Algorithm 4, since $\mathit{inf} \in \mathsf{R}(\mathsf{Closure})$ and $\mathit{inf}.\mathsf{conclusion}.\mathsf{partition} \notin \mathsf{Broken}$, we must have $\mathit{inf}.\mathsf{conclusion} \in \mathsf{Closure}$, and thus $\mathit{inf}.\mathsf{conclusion} \in \mathsf{Closure}_1$.

If $\mathit{inf}.\mathsf{conclusion}.\mathsf{partition} \in \mathsf{Broken}$, there are two cases possible. Either $\mathit{inf} \in \mathsf{R}(\mathsf{Closure}_1)$, thus, $\mathit{inf} \in \mathsf{R}\langle\mathsf{Broken}\rangle(\mathsf{Closure})$. In this case inf is applied in Algorithm 4 during initialization (lines 2–6). Or, otherwise, inf has at least one premise in Repaired, and hence, it is applied in the main loop of Algorithm 4 (lines 7–16). In both cases the algorithm ensures that $\mathit{inf}.\mathsf{conclusion} \in \mathsf{Repaired} \cup \mathsf{Todo}$.

Now, since $\mathsf{Closure}_1 \cup \mathsf{Repaired} \cup \mathsf{Todo}$ is closed under $\mathsf{R}(\mathsf{Closure}_1 \cup \mathsf{Repaired})$ and $\mathsf{Closure} \cap \mathsf{Todo} = \emptyset$, it is also closed under $\mathsf{R}(\mathsf{Closure})$ (if $\mathit{inf} \in \mathsf{R}(\mathsf{Closure})$ is applicable to $\mathsf{Closure}_1 \cup \mathsf{Repaired} \cup \mathsf{Todo}$ then $\mathit{inf} \in \mathsf{R}(\mathsf{Closure}_1 \cup \mathsf{Repaired})$). Since $\mathsf{Closure}'$ is the closure under $\mathsf{R}(\mathsf{Closure})$, we therefore, have $\mathsf{Closure}' = \mathsf{Closure}_1 \cup \mathsf{Repaired} \cup \mathsf{Todo} \subseteq \mathsf{Closure} \cup \mathsf{Todo}$. Hence, $\mathsf{Closure}' \setminus \mathsf{Closure} \subseteq \mathsf{Todo}$, as required. □

After computing the repair Todo of the set Closure using Algorithm 4, we can compute the rest of the closure as in Algorithm 2 using the partially initialized Todo. The correctness of the complete incremental procedure follows from Lemma 1, Lemma 2, and the correctness of our modification of Algorithm 2 when Todo is initialized with missing conclusions of $\mathsf{R}(\mathsf{Closure})$.

Algorithm 4 does not impose any restrictions on the assignment of partitions to expressions. Its performance in terms of the number of operations, however, can substantially depend on this assignment. If we assign, for example, the same partition to all expressions, then in the main loop (lines 7–16) we have to re-apply all inferences in $\mathsf{R}(\mathsf{Closure})$. Thus, it is beneficial to have many different partitions. At another extreme, if we assign a unique partition to every expression, then $\mathsf{R}\langle\mathsf{Broken}\rangle$ would consist of all inferences producing the deleted expressions, and we face the problem of identifying such inferences in lines 2–6. Next, we present a specific partition assignment for the $\mathcal{EL}^+$ rules in Figure 1, which circumvents both of these problems.

5 Incremental Reasoning in $\mathcal{EL}^+$

In this section, we apply our method for updating the classification of $\mathcal{EL}^+$ ontologies computed using the rules in Figure 1. We only consider changes in concept inclusion axioms while resorting to full classification for changes in role inclusions and compositions. We first describe our strategy of partitioning the derived subsumptions, then discuss some issues related to optimizations, and, finally, present an empirical evaluation measuring the performance of our incremental procedure on existing ontologies.

5.1 Partitioning of Derived $\mathcal{EL}^+$ Subsumptions

The inferences R in Figure 1 operate with concept subsumptions of the form $C \sqsubseteq D$. We partition them into sets of subsumptions having the same left-hand side. Formally, the set of partition identifiers $\mathbf{Pts}$ is the set of all $\mathcal{EL}^+$ concepts, and every subsumption $C \sqsubseteq D$ is assigned to the partition corresponding to its left-hand side C. This assignment provides sufficiently many different partitions, which could be as many as there are concepts in the input ontology. It also has the advantage that the inferences $\mathsf{R}\langle \mathit{Pts} \rangle$ for any set *Pts* of partitions can be easily identified. Indeed, note that every conclusion of a rule in Figure 1, except for the initialization rules $\mathbf{R_0}$ and $\mathbf{R}_\top$, has the same left-hand side as one of the premises of the rule. Therefore, $\mathsf{R}\langle \mathit{Pts} \rangle$ can only contain those initialization inferences in $\mathbf{R_0}$ and $\mathbf{R}_\top$ for which $C \in \mathit{Pts}$.

5.2 Optimizations

Let us discuss a few optimizations that are specific to the $\mathcal{EL}^+$ inference rules.

Rule Optimizations: The approach described in Section 4 can be used with any $\mathcal{EL}^+$ classification procedure that implements the inference rules in Figure 1 as they are. Existing implementations, however, include several optimizations to avoid unnecessary applications of some rules. One of such optimizations in ELK prevents applying rule $\mathbf{R}_\sqcap^-$ to conclusions of $\mathbf{R}_\sqcap^+$, and rules $\mathbf{R}_\exists$ and $\mathbf{R}_\circ$ if its left premise was obtained by $\mathbf{R}_\exists$ [15]. Even though the closure computed by Algorithm 1 does not change under such optimizations (the algorithm just derives fewer duplicate conclusions), if the same optimizations are used for deletions in Algorithm 3, some subsumptions that are no longer derivable may remain in Closure. Intuitively, this happens because the inferences for deleting conclusions in Algorithm 3 can be applied in a different order than they were applied in Algorithm 1 for deriving these conclusions. Please refer to the technical report [16] for an extended example of this situation.

To fix this problem, we do not use rule optimizations for deletions in Algorithm 3. To repair the closure using Algorithm 4, we also need to avoid optimizations to make sure that all expressions in broken partitions of Closure are encountered, but it is sufficient to insert only conclusions of optimized inferences into Todo.

Subsumptions That Cannot Be Re-Derived: When Algorithm 3 deletes an expression *exp* from Closure, we mark *exp*.partition as broken because this expression could be re-derived. In some situations this is not possible. One property of the $\mathcal{EL}_\bot^+$ rules in Figure 1, is that they derive only subsumptions of the form $C \sqsubseteq D$ or $C \sqsubseteq \exists R.D$ where C and D occur in the ontology. So, if a deleted subsumption is not of this form for the ontology after deletion, we know that it cannot be re-derived. For example, consider the following ontology $\mathcal{O}$: (ax1) $A \sqsubseteq B$, (ax2) $B \sqsubseteq C$, from which (ax2) is deleted. When the previously derived conclusion $A \sqsubseteq C$ is deleted, there is no need to mark the partition of A as broken since C does not occur in the ontology after the deletion.

Structural Rules: When we apply our incremental procedure for the $\mathcal{EL}_\bot^+$ in Figure 1, we take R^- to be the inferences that are no longer valid after deletion of axioms. An inference by a rule in Figure 1 is not valid when its side condition is not satisfied. For example, for the rule $\mathbf{R}_\sqsubseteq$, the subsumption $D \sqsubseteq E$ may be removed from the ontology,

or for the rule $\mathbf{R}^+_\sqcap$, the conjunction $D_1 \sqcap D_2$ does not occur in the ontology any more. But the impact of these two inferences is different: the conclusion of $\mathbf{R}_\sqsubseteq$ may be not correct if the side condition does not hold, but the conclusion of $\mathbf{R}^+_\sqcap$ is always correct, but may be just irrelevant. This distinction between the rules can be used in our next optimization. We call the rules $\mathbf{R_0}$, $\mathbf{R}_\top$, $\mathbf{R}^+_\sqcap$, $\mathbf{R}^-_\sqcap$, and $\mathbf{R}_\exists$ *structural*—these rules use only the structure of the concepts; they are sound even if their side conditions are not satisfied. Avoiding application of some structural rules during deletions may result in fewer broken partitions as shown in the next example.

Consider an ontology $\mathcal{O}$: (ax1) $A \sqsubseteq B$, (ax2) $A \sqsubseteq C$, (ax3) $(B \sqcap C) \sqcap D \sqsubseteq E$. The rules in Figure 1 derive the following conclusions (with the partition A):

$$A \sqsubseteq A \qquad \text{by } \mathbf{R_0} \text{ since } A \text{ occurs in } \mathcal{O}, \tag{20}$$

$$A \sqsubseteq B \qquad \text{by } \mathbf{R}_\sqsubseteq \text{ to (20) using (ax1)}, \tag{21}$$

$$A \sqsubseteq C \qquad \text{by } \mathbf{R}_\sqsubseteq \text{ to (20) using (ax2)}, \tag{22}$$

$$A \sqsubseteq B \sqcap C \qquad \text{by } \mathbf{R}^+_\sqcap \text{ to (21) and (22) using (ax3)}. \tag{23}$$

Now, assume that (ax3) is deleted from $\mathcal{O}$. Normally, we should revert the inference producing (23) by $\mathbf{R}^+_\sqcap$ using (ax3) in the deletion stage, which would then mark the partition of A as broken. We can, however, leave this rule applied (because it is still sound), which not only makes the partition of A unaffected, but also prevents further deletion of subsumptions $A \sqsubseteq B$ and $A \sqsubseteq C$ by rule $\mathbf{R}^-_\sqcap$ applied to (23).

5.3 Experimental Evaluation

We have implemented the procedure described in Section 4.3 in the OWL EL reasoner ELK v.0.4.0,[1] and performed some experiments to evaluate its performance.

We used three large OWL EL ontologies which are frequently used in evaluations of $\mathcal{EL}$ reasoners [3–6]: the Gene Ontology GO [17] with $84,955$ axioms, an $\mathcal{EL}^+$-restricted version of the GALEN ontology with $36,547$ axioms,[2] and the official January 2013 release of SNOMED CT with $296,529$ axioms.[3]

The recent change history of GO is readily available from the public repository.[4] We took the last (as of April 2013) 342 changes of GO (the first at r560 with $74,708$ axioms and the last at r7991 with $84,955$ axioms). Each change is represented as sets of added and deleted axioms (an axiom modification counts as one deletion plus one addition). Out of the 9 role axioms in GO, none was modified. Unfortunately, similar data was not available for GALEN or SNOMED CT. We used the approach of Cuenca Grau *et.al* [11] to generate 250 versions of each ontology with n random additions and deletions ($n = 1, 10, 100$). For each change history, we classified the first version of the ontology and then classified the remaining versions incrementally. We used a PC with Intel Core i5-2520M 2.50GHz CPU, running Java 1.6 with 4GB of RAM available to JVM.

[1] In fact, the incremental procedure in ELK supports many other features outside of $\mathcal{EL}^+$, such as assertions, disjointness axioms, and restricted use of nominals and datatype restrictions, see `http://elk.semanticweb.org` for the full release notes.

[2] `http://www.co-ode.org/galen/`

[3] `http://www.ihtsdo.org/snomed-ct/`

[4] `svn://ext.geneontology.org/trunk/ontology/`

Table 2. Number of inferences and running times (in ms.) for test ontologies. The results for each incremental stage are averaged (for GO the results are only averaged over changes with a non-empty set of deleted axioms). Time (resp. number of inferences) for initial classification is: GO (r560): 543 (2,224,812); GALEN: 648 (2,017,601); SNOMED CT: 10,133 (24,257,209).

Ontology	Changes	Deletion			Repair		Addition		Total	
	add.+del.	# infer.	$\lvert\mathsf{Broken}\rvert$	time	# infer.	time	# infer.	time	# infer.	time
GO (r560)	84+26	62,384	560	48	17,628	8	58,933	66	138,945	134
GALEN	1+1	3,444	36	18	4,321	4	3,055	13	10,820	39
($\mathcal{EL}^+$ version)	10+10	68,794	473	66	37,583	17	49,662	52	156,039	147
	100+100	594,420	4,508	214	314,666	96	426,462	168	1,335,548	515
SNOMED CT	1+1	4,022	64	120	423	1	2,886	68	7,331	232
(Jan 2013)	10+10	42,026	251	420	8,343	4	31,966	349	82,335	789
	100+100	564,004	3,577	662	138,633	56	414,255	545	1,116,892	1,376

The results of the initial and incremental classifications are given in Table 2. For GO we have only included results for changes that involve deletions (otherwise the averages for deletion and repair would be artificially lower). First note that in each case, the incremental procedure makes substantially fewer inferences and takes less time than the initial classification. Unsurprisingly, the difference is most pronounced for larger ontologies and smaller values of n. Also note that the number of inferences in each stage and the number of partitions $\lvert\mathsf{Broken}\rvert$ affected by deletions, depend almost linearly on n, but not the running times. This is because applying several inferences at once is more efficient than separately. Finally, observe that the repair stage takes a relatively small fraction of the total time.

In order to compare our method to the module-based approach of [11] (the only implemented incremental reasoning procedure for DLs which works for TBox additions and deletions that we are aware of) we classified the same history of GO changes using the implementation included in the standard distribution of Pellet 2.3.2.[5] Pellet provides a consequence-based procedure for $\mathcal{EL}$ classification which was used for re-classifying the affected parts of the ontology. Unfortunately the same experiment was not possible for the other two ontologies due to time-outs (10 hours). The results for GO are as follows: initial classification together with module extraction takes 126 seconds, the average incremental classification 101 seconds, the average numbers of re-computed modules are 634 (when processing deletions) and 672 (for additions).

Abstracting from the much worse time results,[6] which are likely due to a naive implementation of the module-based incremental procedure and/or the $\mathcal{EL}$ algorithm in Pellet, it is interesting to compare the average number of modules which are re-computed during the deletion stage with the average number of broken partitions reported by our algorithm. Intuitively, both of these metrics characterise the number of named concepts for which subsumers need to be re-computed upon an axiom change. The number of modules (634) is greater than the number of broken partitions (560). Interestingly, this relationship is of general nature. We prove in the technical report [16] that if a

[5] `http://clarkparsia.com/pellet/`

[6] Note that the times in Table 2 are in milliseconds, not in seconds.

subsumption $C \sqsubseteq D$ is deleted by our (optimized) incremental algorithm as a result of deleting some axiom α, then α is contained in the locality-based module for C and thus the module must be re-computed. Simply put, the generic module-based approach may not incur less overhead than our method for $\mathcal{EL}^+$.

In general, this relationship does not hold in the other direction since modules can contain more axioms than used in derivations. For example, consider the ontology $\mathcal{O}$ containing $A \sqsubseteq \exists R.B$ and $B \sqsubseteq C$. The rules in Figure 1 derive only $A \sqsubseteq A$ and $A \sqsubseteq \exists R.B$ in the partition for A, thus removing $B \sqsubseteq C$ will not break the partition for A. On the other hand, the locality-based module for A contains all axioms in $\mathcal{O}$, and thus, it has to be re-computed after the deletion. The difference between the number of re-extracted modules and the number of broken partitions is likely to be greater for more complex ontologies, e.g., GALEN. The structure of the anatomical part of GALEN is known to induce very large locality-based modules [11].

Finally, we have evaluated the effectiveness of the two optimizations from Section 5.2 that can reduce the set of broken partitions when some concepts get deleted from the ontology. Avoiding applications of structural rules during deletion gives the most improvement. It reduces the set Broken by roughly 10%, e.g., 498 vs 560 on average for GO. This leads to reduction of the total number of rule applications also by 10%. The time difference is most visible for smaller change sizes, e.g. ± 1 and ± 10 for GALEN and SNOMED CT. Please see the technical report [16] for detailed results.

6 Summary and Future Research

In this paper we have presented a new method for incremental classification of $\mathcal{EL}^+$ ontologies. It is simple, supports both additions and deletions, and does not require deep modification of the base reasoning procedure. Our experiments, though being preliminary due to the shortage of revision histories for real-life $\mathcal{EL}$ ontologies, demonstrate that the reasoning results can be obtained almost instantly after small changes. Potential applications of the method range from background classification of ontologies in editors to stream reasoning and query answering. The method could also be used to handle ABox changes (via a TBox encoding) or easily extended to consequence-based reasoning procedures for more expressive Description Logics [18, 19].

The main idea of our method is that we can benefit from knowing the exact rules of $\mathcal{EL}^+$, which is not possible in the general DRed setting. In particular, we can exploit the 'granularity' of the $\mathcal{EL}^+$ procedure, namely that subsumers of different concepts can be often computed independently of each other. A similar property is a corner stone for the concurrent $\mathcal{EL}$ classification algorithm used in ELK where contexts are similar to our partitions [4]. In the future, we intend to further exploit this property for on-demand proof generation (for explanation and debugging) and distributed $\mathcal{EL}$ reasoning.

References

1. Baader, F., Brandt, S., Lutz, C.: Pushing the $\mathcal{EL}$ envelope. In: Kaelbling, L., Saffiotti, A. (eds.) Proc. 19th Int. Joint Conf. on Artificial Intelligence (IJCAI 2005), pp. 364–369. Professional Book Center (2005)

2. Motik, B., Cuenca Grau, B., Horrocks, I., Wu, Z., Fokoue, A., Lutz, C. (eds.): OWL 2 Web Ontology Language: Profiles. W3C Recommendation (October 27, 2009), http://www.w3.org/TR/owl2-profiles/
3. Baader, F., Lutz, C., Suntisrivaraporn, B.: Efficient reasoning in $\mathcal{EL}^+$. In: Parsia, B., Sattler, U., Toman, D. (eds.) Proc. 19th Int. Workshop on Description Logics (DL 2006). CEUR Workshop Proceedings, vol. 189, CEUR-WS.org (2006)
4. Kazakov, Y., Krötzsch, M., Simančík, F.: Concurrent classification of $\mathcal{EL}$ ontologies. In: Aroyo, L., Welty, C., Alani, H., Taylor, J., Bernstein, A., Kagal, L., Noy, N., Blomqvist, E. (eds.) ISWC 2011, Part I. LNCS, vol. 7031, pp. 305–320. Springer, Heidelberg (2011)
5. Mendez, J., Ecke, A., Turhan, A.Y.: Implementing completion-based inferences for the $\mathcal{EL}$-family. In: Rosati, R., Rudolph, S., Zakharyaschev, M. (eds.) Proc. 24th Int. Workshop on Description Logics (DL 2011). CEUR Workshop Proceedings, vol. 745, pp. 334–344. CEUR-WS.org (2011)
6. Lawley, M.J., Bousquet, C.: Fast classification in Protégé: Snorocket as an OWL 2 EL reasoner. In: Proc. 6th Australasian Ontology Workshop (IAOA 2010), pp. 45–49 (2010)
7. Schulz, S., Cornet, R., Spackman, K.A.: Consolidating SNOMED CT's ontological commitment. Applied Ontology 6(1), 1–11 (2011)
8. Suntisrivaraporn, B.: Module extraction and incremental classification: A pragmatic approach for ontologies. In: Bechhofer, S., Hauswirth, M., Hoffmann, J., Koubarakis, M. (eds.) ESWC 2008. LNCS, vol. 5021, pp. 230–244. Springer, Heidelberg (2008)
9. Sirin, E., Parsia, B., Cuenca Grau, B., Kalyanpur, A., Katz, Y.: Pellet: A practical OWL-DL reasoner. J. of Web Semantics 5(2), 51–53 (2007)
10. Halashek-Wiener, C., Parsia, B., Sirin, E.: Description logic reasoning with syntactic updates. In: Meersman, R., Tari, Z. (eds.) OTM 2006. LNCS, vol. 4275, pp. 722–737. Springer, Heidelberg (2006)
11. Cuenca Grau, B., Halaschek-Wiener, C., Kazakov, Y., Suntisrivaraporn, B.: Incremental classification of description logics ontologies. J. of Automated Reasoning 44(4), 337–369 (2010)
12. Gupta, A., Mumick, I.S., Subrahmanian, V.S.: Maintaining views incrementally. In: Buneman, P., Jajodia, S. (eds.) Proc. 1993 ACM SIGMOD Int. Conf. on Management of Data, May 26-28, pp. 157–166. ACM Press, Washington, D.C. (1993)
13. Volz, R., Staab, S., Motik, B.: Incrementally maintaining materializations of ontologies stored in logic databases. J. of Data Semantics 2, 1–34 (2005)
14. Barbieri, D.F., Braga, D., Ceri, S., Della Valle, E., Grossniklaus, M.: Incremental reasoning on streams and rich background knowledge. In: Aroyo, L., Antoniou, G., Hyvönen, E., ten Teije, A., Stuckenschmidt, H., Cabral, L., Tudorache, T. (eds.) ESWC 2010, Part I. LNCS, vol. 6088, pp. 1–15. Springer, Heidelberg (2010)
15. Kazakov, Y., Krötzsch, M., Simančík, F.: ELK: a reasoner for OWL EL ontologies. Technical report, University of Oxford (2012), http://elk.semanticweb.org
16. Kazakov, Y., Klinov, P.: Incremental classification for OWL EL without bookkeeping. Technical report, University of Ulm (2013), http://elk.semanticweb.org
17. Mungall, C.J., Bada, M., Berardini, T.Z., Deegan, J.I., Ireland, A., Harris, M.A., Hill, D.P., Lomax, J.: Cross-product extensions of the gene ontology. J. of Biomedical Informatics 44(1), 80–86 (2011)
18. Kazakov, Y.: Consequence-driven reasoning for Horn $\mathcal{SHIQ}$ ontologies. In: Boutilier, C. (ed.) Proc. 21st Int. Joint Conf. on Artificial Intelligence (IJCAI 2009), pp. 2040–2045 (2009)
19. Simančík, F., Kazakov, Y., Horrocks, I.: Consequence-based reasoning beyond Horn ontologies. In: Walsh, T. (ed.) Proc. 22nd Int. Joint Conf. on Artificial Intelligence (IJCAI 2011), pp. 1093–1098. AAAI Press/IJCAI (2011)

Secure Manipulation of Linked Data

Sabrina Kirrane[1,2], Ahmed Abdelrahman[1], Alessandra Mileo[1], and Stefan Decker[1]

[1] Digital Enterprise Research Institute
National University of Ireland, Galway
{firstname.lastname}@deri.ie
http://www.deri.ie
[2] Storm Technology, Ireland
http://www.storm.ie

Abstract. When it comes to publishing data on the web, the level of access control required (if any) is highly dependent on the type of content exposed. Up until now RDF data publishers have focused on exposing and linking public data. With the advent of SPARQL 1.1, the linked data infrastructure can be used, not only as a means of publishing open data but also, as a general mechanism for managing distributed graph data. However, such a decentralised architecture brings with it a number of additional challenges with respect to both data security and integrity. In this paper, we propose a general authorisation framework that can be used to deliver dynamic query results based on user credentials and to cater for the secure manipulation of linked data. Specifically we describe how graph patterns, propagation rules, conflict resolution policies and integrity constraints can together be used to specify and enforce consistent access control policies.

1 Introduction

In the early days, the Web was primarily used as a medium for sharing and linking static information. However it wasn't until challenges with respect to data confidentiality, authenticity and integrity were addressed that electronic business became common place. It is not surprising that the Semantic Web is following a similar evolution. With the advent of SPARQL 1.1, an update language for RDF graphs, it is possible for the Semantic Web to evolve from a medium for publishing and linking data to a dynamic read/write distributed data source, that can support the next generation of electronic business applications. However, in order to make the move from simply exposing to maintaining linked data we must first provide solutions for data security and integrity.

To date researchers have focused primarily on the specification of access control policies for RDF stores based on RDF patterns [13, 8, 4, 1, 6] or the specification and enforcement of access control ontologies over linked data [3, 14]. Although some of these authors touch upon reasoning over access control policies, they do not propose a general authorisation framework which can support

H. Alani et al. (Eds.): ISWC 2013, Part I, LNCS 8218, pp. 248–263, 2013.

reasoning based on a combination of propagation rules, conflict resolution policies and integrity constraints.

In previous work, we provided a summary of access control requirements that are needed to cater for Discretionary Access Control (DAC) over RDF data [11]. In this paper, we demonstrate how authorisations together with stratified Datalog rules can be used to enforce DAC over the RDF data model. The contributions of the paper can be summarised as follows: We (i) demonstrate how the hierarchical Flexible Authorisation Framework [9] can be adapted to work with graph data; (ii) provide a formal definition of an RDF instantiation of the framework, which we refer to as the "Graph based Flexible Authorisation Framework" or G-FAF; (iii) describe how together pattern matching and propagation rules can be used to ease the maintenance of access control policies for linked data sources; and (iv) show how conflict resolution policies and integrity constraints can ensure access control policy integrity.

The remainder of the paper is structured as follows: Section 2, examines alternative approaches for the enforcement and administration of access control over RDF data. Section 3, provides an overview of DAC requirements in the context of the RDF data model and describes the Flexible Authorisation Framework, which has been successfully applied to both the relational and the xml data models. Section 4, demonstrates how the authorisation framework can be extended to cater for the RDF graph data model. Section 5, details how graph patterns, propagation rules, integrity constraints and conflict resolution policies can be used to specify and enforce access control over the RDF data model. Whereas Section 6, discusses how the extended framework can be used to enforce access control over linked data sources and details the results of our performance evaluation. Finally Section 7, summarises the contributions and outlines directions for future work.

2 Related Work

Initially Semantic Web researchers focused on the modelling and the enforcement of access control over RDF stores. A number of authors have proposed access control policies based on RDF patterns that can be mapped to one or more RDF triples [13, 8, 4, 1]. Reddivari et al. [13] define a set of actions required to manage an RDF store and demonstrate how query based access control can be used to permit or prohibit access based on these actions. The authors propose default and conflict preferences that can simply be set to either permit or deny. Jain and Farkas [8] propose a data level security model which can be used to protect both explicit and inferred triples. They provide formal definitions for a number of RDF security objects and define an algorithm which generates security labels, based on a security policy and a conflict resolution strategy. Limited details of the implementation are supplied and no evaluation is performed. Whereas Abel et al. [1] propose the evaluation of access control policies at both the query and the data layers. Access conditions that are not dependent on RDF data are evaluated by a policy engine. Whereas access conditions that are dependent on

RDF data are injected into the query. Such an approach requires the substitution of variables to ensure uniqueness however in doing so they are able to leverage the highly optimized query evaluation features of the RDF store. The authors adopt a denial by default conflict resolution strategy.

Gabillon and Letouzey [6] highlight the possible administration burden associated with the maintenance of access control policies that are based on triple patterns. They propose the logical distribution of RDF data into SPARQL views and the subsequent specification of access control policies based on existing RDF graphs or predefined views. They describe a query based enforcement framework whereby each user defines a security policy for the RDF graphs/views that they own. The authors acknowledge the need for conflict resolution however, they do not propose a conflict resolution strategy.

More recently the focus has shifted to the specification and enforcement of access control policies over web resources. Costabello et al. [3] and Sacco et al. [14] both propose access control ontologies and enforcement frameworks that rely on SPARQL ASK queries to determine if the requester possesses the attributes necessary to access the requested resource. Costabello et al. [3] use context data supplied by the requester to limit the scope of the SPARQL query to authorised named graphs. The authors propose the disjunctive evaluation of policies thus circumventing the need for a conflict resolution mechanism. Whereas Sacco et al. [14] provide a filtered view of a data providers FOAF profile based on a matching between the data providers privacy preferences and the requesters attributes. Policies can be specified for an entire graph, one or more triples or individual subjects, predicates and objects. The authors do not propose any conflict resolution strategy.

In our early work, we demonstrated how annotated RDF can be used to limit access to triples and to derive access rights for inferred triples using annotated RDFS inference rules [12]. In this paper, we allow for the specification of authorisations based on quad patterns, thus catering multiple levels of granularity (i.e. one or more graphs, triples, classes or properties). Moreover, we provide a general mechanism for the administration and enforcement of access control policies using a combination of propagation rules, integrity constraints and conflict resolution policies.

3 Preliminaries

In previous work [11], we examined how DAC principles, that have been successfully applied to relational and XML data, can be applied to the RDF data model. In this paper, we introduce the hierarchical *Flexible Authorisation Framework* [9], henceforth referred to as H-FAF, and demonstrate how it can be extended to cater for DAC over the RDF graph data model, which we intuitively name G-FAF. We start by providing a summary of DAC requirements for the RDF data model, before providing the necessary background information about the H-FAF data system and authorisation architecture.

3.1 Discretionary Access Control for RDF Data

In DAC access to resources is constrained by a central access control policy however, users are allowed to override the central policy by passing their access rights on to others [16], known in the literature as delegation. DAC principles that have been successfully integrated into a number of operating systems, databases and information systems developed by well known software vendors (e.g. Oracle, Microsoft, SAP, IBM). However, our decision to base our work on the DAC model was threefold: it has been adopted by several relational DBMS vendors; its inherent flexibility makes it particularly suitable for distributed data; and its potential for handling context based authorisations in the future. Based on our analysis, of DAC for both the relational and XML data models [11], an authorisation framework needs to be able to cater the following requirements:

- In order to ensure the expressivity and the maintainability of access control policies it should be feasible to specify *authorisations* at multiple levels of granularity, from both a data (i.e. nodes, arcs, triples, collection of triples and name graphs) and a schema (i.e. classes and properties) perspective.
- Like the relational and XML data models RDF access rights should be tightly coupled with the operations performed on the data model. However, as graph update operations can only be applied to triples and graph management operations are only appropriate for graphs, *integrity constraints* are needed to ensure the consistency of the access control policies.
- In both the relational and hierarchical data models authorisations can be derived based on the schema. When it comes to the RDF data model similar *derivations* are highly desirable as they simplify authorisation maintenance.
- In DAC access to resources is constrained by a central access control policy however, users are permitted to pass their own access rights on to others [16], known formally as *delegation*.
- As conflicts can occur as a result of inconsistent explicit, derived and delegated policies *conflict resolution* strategies are required to ensure a conclusion can always be reached. Samarati [15] highlights the need for a flexible conflict resolution mechanism which can support different *conflict resolution* strategies depending on the situation.

3.2 H-FAF Data System and Authorisation Framework

The H-FAF is an authorisation framework that can be used to restrict access to different classes of data objects (e.g. files, relations, objects, images), with different access control requirements. The authors provide a general definition for a data system and devise a modular architecture which together with declarative rules can be used to ease access control policy administration, by exploiting the hierarchical structure of the data system components.

Data System Components. An authorisation framework describes the items to be protected (`data items`), to whom access is granted (users, groups, roles collectively known as `authorisation subjects`) and the operations that need

to be protected (access rights). Together these components are known as a data system formally defined by Jajodia and Samarati [9] as follows:

Definition 1 (Data System). A *Data System (DS)* is a 5-tuple $\langle OTH, UGH, RH, A, Rel \rangle$ where: OTH is an *object-type hierarchy*; UGH is a *user-group hierarchy*; RH is a *role-hierarchy*; A is a set of *access rights* and Rel is a set of n-ary *relationships* over the different elements of DS.

Authorisation Framework. In addition to the formal data system definition Jajodia and Samarati [9] propose a number of distinct components that together provide support for access control enforcement and administration:

- *Authorisations* are rules that dictate the access rights that authorisation subjects are allowed/prohibited to perform on data items.
- *Propagation Rules* enable the derivation of implicit authorisations from explicit authorisations and the hierarchical structure of data system components.
- *Conflict Resolution Policies* are rules that provide flexible support for different conflict resolution strategies.
- *Integrity Constraints* are rules that enforce restrictions on authorisation specification thus decreasing the potential for runtime errors.

Rules are expressed in stratified Datalog with negation and are constructed from a combination of explicit authorisations, historical authorisations and both the hierarchical structure of and the relationship between the different data system components.

4 From a Hierarchical to a Graph Data System

As both the hierarchical structure of and the relationship between the data system components can be recorded as RDF in this paper we adapt and extend the original data system and rule definitions to work with the RDF data model. As per the original framework we chose a declarative approach as it has been proven to work well and is based on familiar concepts. We start by describing the individual G-FAF data system components in the context of RDF and extend the original formal definition of a data system to cater for graph data structures. Although we are dealing with graph data the authorisations and the rules can also be expressed using stratified Datalog with negation. Throughout the paper, we use examples from the Berlin SPARQL Benchmark (BSBM) Dataset [1] as it is a well known dataset which is sufficiently complex to represent a real world use case. Prefixes are used as a shorthand notation for each vocabulary (e.g. rdf, rdfs, bsbm) and variables are represented using a ? prefix. The following default prefix is used to increase readability:
(http://www4.wiwiss.fu-berlin.de/bizer/bsbm/v01/instances/dataFromVendor1).

[1] Berlin SPARQL Benchmark (BSBM) - Dataset Specification, http://wifo5-03.informatik.uni-mannheim.de/bizer/berlinsparqlbenchmark/spec/Dataset/

4.1 Individual Data System Components

In G-FAF `data items`, `access rights` and `authorisation subjects` are represented as one or more graphs that may or may not be disjoint.

Data Items. In the Semantic Web information is represented as *RDF triples* that are used to make statements about resources in the form of subject-predicate-object expressions. An *RDF graph* is a finite set of RDF triples. *Named graphs* are used to collectively refer to a number of RDF statements. Although there are several RDF representation formats in this paper we use *nquads*.

Definition 2 (RDF Quad). An *RDF Quad* is formally defined as a 4-tuple $\langle S, P, O, G \rangle \in \mathbf{UB} \times \mathbf{U} \times \mathbf{UBL} \times \mathbf{U}$ [2], where S is called the *subject*, P the *predicate*, O the *object* and G the *named graph*. **U**, **B** and **L**, are in turn used to represent *URIs*, *blank nodes* and *literals* respectively.

Example 1 (RDF Quad). The following quad states that there exists a triple in the Graph2013 dataset stating that Vendor1 is a vendor.

```
:Vendor1 rdf:type bsbm:Vendor :Graph2013
```
□

Access Rights. Like databases and file systems access can be restricted based on the operations that a user attempts to execute on the data items [11]. In the case of RDF these operations take the form of: graph query operations (`SELECT`, `CONSTRUCT`, `ASK` and `DESCRIBE`); graph update operations (`INSERT`, `DELETE`, `DELETE/INSERT`); and a number of operations specifically for graph management (`DROP`, `COPY`, `MOVE` and `ADD`). Three additional access rights are required to facilitate access control administration, namely: `GRANT`, `REVOKE` and `FULL ACCESS`. The `GRANT` privilege allows users to grant access to others based on their own privileges. Whereas the `REVOKE` privilege allows users to revoke the access rights they have granted to others. `FULL ACCESS` is a super access right that subsumes all other access rights. We model the operations as one or more RDF graphs and use vocabularies such as RDFS to define a partial order over the operations. Although it is possible to infer implicit access rights based on the partial order, we do not provide specific details in this paper.

Authorisation Subjects. Subject is an umbrella term used to collectively refer to different user credentials. We propose the verification of access based on credential matching, as such we make no distinction between a user *playing* a role as opposed to *belonging* to a group. Therefore, we merge both the user-group and role hierarchies and refer to them simply as `authorisation subjects`. Such a merge does not impact the specification or enforcement of authorisations and in fact affords a greater degree of flexibility with respect to the inclusion of additional types of user credentials. As RDF is a web based distributed data model we extend the subject definition, to include user attributes. Combined users, groups, roles and attributes can be represented as one or more RDF graphs possibly disjoint.

[2] For conciseness, we represent the union of sets simply by concatenating their names.

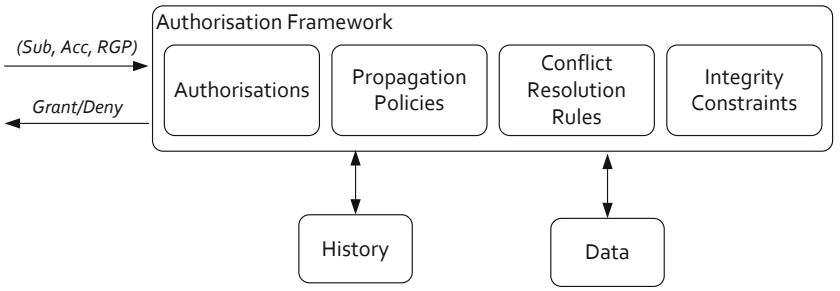


Fig. 1. Authorisation Framework

4.2 Extending the Original Data System Definition

We formally extend the original definition of a data system to consider components that are represented as graphs as opposed to hierarchies. As the relationship between `data items`, `access rights` and `authorisation subjects` can also be represented as RDF it is is not necessary to define a set of relations over the different elements of the data system. Although in this paper we focus specifically on RDF the extended data system definition is more general than the original and therefore it can be applied to both hierarchical and graph data models.

Definition 3 (Graph Data System). A *Graph Data System (GDS)* is defined as a 3-tuple $\langle DIG, ASG, ARG \rangle$ where: DIG represents one or more *data graphs*, that may be disjoint; ASG denotes one or more *subject graphs*; and ARG stands for graphs of *access rights* used to restrict access to the data items in DIG.

5 G-FAF Authorisation Enforcement and Administration

Given an arbitary but fixed `Graph Data System`, we describe the individual G-FAF components (Fig. 1) and demonstrate how together these components can be used to deliver dynamic query results based on user credentials and to cater for the secure manipulation of RDF graph data (Section. 6). We extend the original framework to include the *Data* component, which is necessary to infer new access control policies based on a combination of RDF data and rules. Although in this paper, we do not examine the role of the *History* component, it is worth noting that historical information is important for accountability and also to cater for contextual access control policies that rely on historical data.

5.1 Authorisations

An *RDF Quad Pattern* is an RDF quad with optionally a variable V in the subject, predicate, object and/or graph position. A *Quad Pattern* is a flexible

mechanism which can be used to grant/restrict access to an RDF quad, a collection of RDF quads (multiple quads that share a common subject), a named graph (arbitrary views of the data), specific classes or properties.

Example 2 (RDF Quad Pattern). A single quad pattern containing variables in both the subject (?S) and graph (?G) positions is used to match products spanning multiple graphs.
?S `rdf:type bsbm:Product` ?G. □

More expressive authorisations can be achieved using an *RDF Graph Pattern*, which is composed of multiple quad patterns.

Example 3 (RDF Graph Pattern). A graph pattern with variables in both the subject (?S) and graph (?G) positions is used to match all products for Producer1 from multiple graphs.
?S `rdf:type bsbm:Product` ?G_1.
?S `bsbm:producer bsbm-inst:Producer1` ?G_2. □

In order to cater for certain conflict resolution strategies and for the delegation of access rights we extend the original authorisation definition to include *Type* and *By* attributes. The *Type* attribute is necessary to differentiate between explicit and inferred authorisations, whereas the *By* attribute is used to denote the person who created the authorisation. By default the *Type* attribute is set to *E* for explicit and the *By* attribute defaults to a reserved literal *OWNER*.

Definition 4 (Authorisation). An authorisation is represented as a 6-tuple $\langle Sub, Acc, Sign, RGP, Type, By \rangle$. *Sub* represents the *authorisation subject*. *Acc* is used to denote *access rights*. *Sign* indicates if the user is *granted* or *denied* access. *RGP* symbolises the *RDF Graph Pattern*. *Type* is used to indicate if the authorisation is explicit (E) or implicit (I) and *By* represents the person who created the authorisation.

Example 4 (Authorisation). Using the following authorisation a bsbm:admin can grant all bsbm:partners `UPDATE` access to all triples in the :Graph2008 graph.
$\langle$ `bsbm:Partner`, `UPDATE`, +, $\langle$?S, ?P, ?O, `:Graph2008`$\rangle$, `E`, `bsbm:Admin` $\rangle$ □

5.2 Propagation Policies

Propagation policies can be used to simplify authorisation administration by allowing for the derivation of implicit authorisations from explicit ones. For example, we can derive new authorisations based on the logical organisation of `authorisation subjects`, `access rights` and `data items` [10] or the RDF Schema vocabulary [11]. We provide a formal definition for a propagation rule which can be used as a blueprint for both general and specific derivation rules (Def. 5). In addition, we present a propagation algorithm (Alg. 1) which can be used to either evaluate the propagation policy at query time or alternatively to materialise implicit authorisations when authorisations are added or removed.

Definition 5 (Propagation Policy). A *Propagation Policy* is a a rule of the following format: $\langle ?Sub_y, ?Acc_y, ?Sign_y, \langle RGP_y \rangle, ?Type_y, ?By_y \rangle \leftarrow$
$\langle ?Sub_x, ?Acc_x, ?Sign_x, \langle RGP_x \rangle, ?Type_x, ?By_x \rangle , [\langle RGP_1 \rangle \wedge ... \wedge \langle RGP_x \rangle \wedge ... \wedge \langle RGP_y \rangle \wedge ... \wedge \langle RGP_n \rangle]$
The premise is composed of an *Authorisation* and an *RDF Graph Pattern*. If the *Authorisation* exists in the list of Authorisations and the *RDF Quad Pattern* exists in the Data then we can infer the conclusion.

Example 5 (Subject Hierarchy Inheritance). Using the following rule the access rights assigned to employees can be derived for all managers.
$\langle$ `bsbm:Mgr`, $?Acc, ?Sign, \langle ?S, ?P, ?O, ?G \rangle, I, ?By \rangle \leftarrow$
$\langle$ `bsbm:Emp`, $?Acc, ?Sign, \langle ?S, ?P, ?O, ?G \rangle, ?Type, ?By \rangle$,
$[\langle$ `bsbm:Mgr`, `rdf:type`, `bsbm:Emp`, $?G \rangle \wedge \langle ?S, ?P, ?O, ?G \rangle]$ □

Example 6 (Class to Instance Propagation). The following rules propagates the access rights assigned to a *bsbm:Product* class to all instances of the class.
$\langle ?Sub, ?Acc, ?Sign, \langle ?Z, ?Y, ?A, ?G_x \rangle, I, ?By \rangle \leftarrow$
$\langle ?Sub, ?Acc, ?Sign, \langle$ `bsbm:Product`, `rdf:type`, `rdf:Class`, $?G_y \rangle, ?Type, ?By \rangle$,
$[\langle ?Z,$ `rdf:type`, `bsbm:Product`, $?G_z \rangle \wedge \langle ?Z, ?Y, ?A, ?G_x \rangle]$

$\langle ?Sub, ?Acc, ?Sign, \langle ?Z, ?Y, ?A, ?G_x \rangle, I, ?By \rangle \leftarrow$
$\langle ?Sub, ?Acc, ?Sign, \langle$ `bsbm:Product`, `rdf:type`, `rdf:Class`, $?G_y \rangle, ?Type, ?By \rangle$,
$[\langle ?Z,$ `rdf:type`, `bsbm:Product`, $?G_z \rangle \wedge \langle ?A, ?Y, ?Z, ?G_x \rangle]$ □

5.3 Conflict Resolution Rules

Rather than propose a conflict resolution strategy we provide a formal definition for a conflict resolution rule (Def. 6) that can be used to determine access given several different conflict resolution strategies. For example conflict resolution policies based on the structure of the `graph data system` components; the sensitivity of the data requested; or contextual conditions pertaining to the requester. In order to cater for type matching the Authorisation *RDF Graph Pattern* is replaced with an *Extended RDF Graph Pattern* which includes the reserved words `CON` and `VAR`, that are used to match all constants and variables respectively. As multiple conflict resolution rules may be applicable, each rule should be assigned a priority and rules should be evaluated based on priority until the conflict has been resolved. The default rule which matches everything is assigned the lowest priority thus ensuring a conclusion can always be drawn.

Definition 6 (Conflict Resolution Rule). A *Conflict Resolution Rule* is a rule of the following format:
$\langle ?Sub_x, ?Acc_x, ?Sign_x, \langle ERGP_x \rangle, ?Type_x, ?By_x \rangle \leftarrow$
$\langle ?Sub_x, ?Acc_x, ?Sign_x, \langle ERGP_x \rangle, ?Type_x, ?By_x \rangle > \langle ?Sub_y, ?Acc_y, ?Sign_y, \langle ERGP_y \rangle, ?Type_y, ?By_y \rangle$
where $>$ indicates the authorisation to the left of the $>$ symbol takes precedence over the authorisation to the right; $?Sub$, $?Acc$, $?Type$ and $?By$ match authorisation *subjects*, *access rights*, *type* and *by* attributes, represented as constants or variables, and $ERGP$ denotes an *Extended RDF Graph Pattern*.

Data: Authorisations, PropPolicy, RDFData
Result: Authorisations
forall the *pol in PropPolicy* **do**
  if *Authorisations CONTAINS pol.premise.authorisation* **then**
    if *RDFData CONTAINS pol.premise.graphPattern* **then**
      Authorisations += pol.conclusion.authorisation
    end
  end
end
return Authorisations

Algorithm 1. Applying the Propagation Rules

Example 7 (Most Specific takes precedence). The following rule states that authorisations assigned to specific subject, predicate and objects in a graph override authorisations assigned to the whole graph.
$\langle ?Sub_x, ?Acc_x, ?Sign_x, \langle \texttt{CON}, \texttt{CON}, \texttt{CON}, \texttt{CON} \rangle, ?Type_x, ?By_x \rangle \leftarrow$
$\langle ?Sub_x, ?Acc_x, ?Sign_x, \langle \texttt{CON}, \texttt{CON}, \texttt{CON}, \texttt{CON} \rangle, ?Type_x, ?By_x \rangle >$
$\langle ?Sub_y, ?Acc_y, ?Sign_y, \langle \texttt{VAR}, \texttt{VAR}, \texttt{VAR}, \texttt{CON} \rangle, ?Type_y, ?By_y \rangle$ □

Example 8 (Explicit overrules Implicit). Using the following rule it is possible to state that explicit authorisations override implicit authorisations.
$\langle ?Sub_x, ?Acc_x, ?Sign_x, \langle ?S_x, ?P_x, ?O_x, ?G_x \rangle, E, ?By_x \rangle \leftarrow$
$\langle ?Sub_x, ?Acc_x, ?Sign_x, \langle ?S_x, ?P_x, ?O_x, ?G_x \rangle, E, ?By_x \rangle >$
$\langle ?Sub_y, ?Acc_y, ?Sign_y, \langle ?S_y, ?P_y, ?O_y, ?G_y \rangle, I, ?By_y \rangle$ □

5.4 Integrity Constraints

Integrity constraints are used to restrict authorisation creation based on the existing relationships between SPARQL operations and RDF data items. For example, `INSERT` and `DELETE` can only be applied to an RDF quad whereas `DROP`, `CREATE`, `COPY`, `MOVE` and `ADD` can only be associated with a named graph. As per conflict resolution rules the integrity constraints use the *Extended RDF Graph Pattern* which includes the reserved words `CON` and `VAR` that are used to match all constants and variables respectively. We provide a formal definition of an integrity constraint (Def. 7) and demonstrate how general rules can be used to constrain the specification of the `INSERT` (Ex. 9) and the `CREATE` (Ex. 10) access rights.

Definition 7 (Integrity Constraint). An *Integrity Constraint* is a rule of the following format:
$\text{error} \leftarrow [\neg]\ \langle ?Sub, ?Acc, ?Sign, \langle ERGP_x \rangle, ?Type, ?By \rangle$
where square brackets [] are used to denote the optional classical negation prefix ($\neg$); $?Sub$, $?Acc$, $?Type$ and $?By$ match authorisation *subjects*, *access rights*, *type* and *by* attributes, represented as constants or variables and $ERGP$ denotes an *Extended RDF Graph Pattern*.

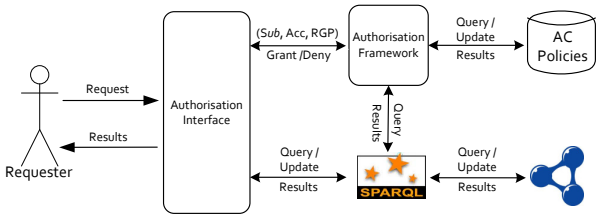


Fig. 2. Authorisation Architecture

Example 9 (INSERT Constraint). Using an integrity constraint we can ensure that the `INSERT` access right is only applied to RDF quads.
error $\leftarrow \neg\langle ?Sub, \texttt{INSERT}, ?Sign, \langle \texttt{CON}, \texttt{CON}, \texttt{CON}, \texttt{CON}\rangle, ?Type, ?By\rangle$ □

Example 10 (CREATE Constraint). The following integrity constraint ensures that the `CREATE` graph management access right is only associated with named graphs.
error $\leftarrow \neg\langle ?Sub, \texttt{CREATE}, ?Sign, \langle \texttt{VAR}, \texttt{VAR}, \texttt{VAR}, \texttt{CON}\rangle, ?Type, ?By\rangle$ □

6 Application and Evaluation

RDF data is mostly exposed on the web via sparql endpoints. Although the architecture we propose will work with any query language in this paper we describe how it can be used in conjunction with SPARQL to enforce and administer access control over RDF. First, we discuss how the framework can be used to enforce and administer access control over linked data sources. Next, we examine the performance of our Java implementation of the framework.

6.1 Applying the Framework to Linked Data

The *Authorisation Architecture* in Fig. 2 depicts how G-FAF can be used for the enforcement and administration of access control policies over linked data sources. We do not focus on authentication in this paper, and thus we assume that the credentials supplied by the requester have been successfully authenticated via alternative means, for example WebId and self-signed certificates, working transparently over HTTPS.

Enforcement of Authorisations. In addition to the usual sparql query the requester must submit their credentials, which are verified by an external authentication system. The *Authorisation Interface* maps the sparql query to an *Authorisation Request* of the form $\langle Sub, Acc, RGP\rangle$ (a subset of Def. 4) and submits it to the *Authorisation Framework* (Fig. 2). The authorisation algorithm (Alg. 2) checks if the *Authorisation Request* can be derived using the *Authorisations* and the *Conflict Resolution Policies*. If the algorithm manages

```
Data: AuthRequest, AuthHashMap, ConflictPolicy
Result: grant/deny
key = AuthRequest.sub + AuthRequest.acc
authHashSet = getAuthHashSet(key, AuthHashMap)
quadHashMap = createQuadHashMap(authHashSet)
dominantAuth = quadHashMap.Auth
while quadHashMap CONTAINS AuthRequest.RGP do
 | authMatches += quadHashMap.Auth
end
if authMatches CONTAINS true and authMatches CONTAINS false then
 | dominantAuth = resolveConflict(authMatches, ConflictPolicy)
end
return dominantAuth.Sign
```

Algorithm 2. Authorisation Enforcement Algorithm

```
Data: AuthRequest, AuthHashMap, IntegrityPolicy, PropRules
Result: true/false
newAuth = AuthRequest + sign.Grant + type.E + by.Owner
if AuthRequest.Acc==INSERT or AuthRequest.Acc==ADD or
AuthRequest.Acc==COPY then
 | if checkIntegrity(AuthRequest, IntegrityPolicy) = true then
 | | AuthHashMap += newAuth
 | | AuthHashMap = applyPropRules(AuthHashMap, PropRules)
 | | return true
 | end
end
else if AuthRequest.Acc==DELETE or AuthRequest.Acc==DROP or
AuthRequest.Acc==MOVE then
 | AuthHashMap -= newAuth
 | return true
end
return false
```

Algorithm 3. Authorisation Administration Algorithm

to successfully derive the authorisation, access to the requested data is granted otherwise the request is denied. If access is granted the *Authorisation Interface* passes the sparql query to the *Query Engine*, which in turn processes the query in the normal way. Finally, the query results are returned to the *Requester* via the *Authorisation Interface*. In the current implementation the `subject` must be granted access to each triple in order to be permitted to execute the query. In future work we plan to investigate the data integrity implications of granting access to subsets of the graph pattern through query rewriting. For example, if a user requests the names of all employees who earn more than 50,000, and that user is denied access to salary data, all employees would be returned leading them to believe that this is the answer to their query.

Table 1. Dataset and Authorisations description

	DS_1	DS_2	DS_3	DS_4	DS_5
quads	250223	500258	1000109	2000164	4000936
scale factor	830	1689	3402	6830	13780
file size (MB)	24.5	49	98	195	391
	QS_1	QS_2	QS_3	QS_4	QS_5
authorisations	60000	120000	240000	480000	960000
file size (MB)	6.5	13	26	53	105

Administration of Authorisations. We propose an ownership model, whereby the data producer is granted `FULL ACCESS` to the data items they create. When a user issues a graph update or graph management query, access is verified using the authorisation algorithm (Alg. 2). If authorisation succeeds the sparql query is passed to the *Query Engine*. For `INSERT`, `ADD` or `COPY` operations, if the query succeeds the administration algorithm (Alg. 3) ensures it adheres to the integrity constraints prior to creating a new authorisation. For `DELETE`, `DROP` or `MOVE` operations, if the query succeeds the administration algorithm (Alg. 3) simply deletes relevant authorisations from the access control policy. In both instances the update of both the RDF graph and the authorisation table should be wrapped in a transaction to ensure that either both or neither succeed.

Delegation of Access Rights. In order to cater for delegation of access control, a number of administration modules are required. For example the ability to list your own access rights, grant/revoke access rights to others and view the access rights you have delegated. Based on the ownership model data producers are granted `FULL ACCESS` to the data items they create and have the ability to `GRANT` and `REVOKE` access to/from others. As neither the grant nor the revoke algorithms are dependent on the data model traditional revocation approaches such as cascading [5, 7] and non-cascading [2] can be used in conjunction with the proposed framework.

6.2 Performance Evaluation

For the evaluation of G-FAF we created three separate experiments to: (i) examine the overhead associated with access control over different data sets; (ii) deduce the impact given an increasing number of authorisations; and (iii) determine the performance increase for a number of propagation rules (the most expensive administration operation). The benchmark system has an Intel(R) Xeon(R) CPU 8 core 2.13GHz processor, 64 GB of memory and runs Debian 6.0.3. The authorisation framework was written in Java and the evaluation was performed over an in memory store using Jena ARQ. Both the datasets (Table. 1) and the queries were generated from the Berlin SPARQL Benchmark (BSBM) dataset. Two separate query sets were created: (i) QS_S which contained *10* SELECT

Table 2. Queries over increasing datasets

		DS_1	DS_2	DS_3	DS_4	DS_5
QS_S *query time (ms)*	$\emptyset$	345	657	1432	2604	5005
	$\exists$	429	700	1164	2549	5149
QS_U *query time (ms)*	$\emptyset$	8	8	9	8	9
	$\exists$	9	9	9	9	9

Table 3. Queries over increasing authorisations

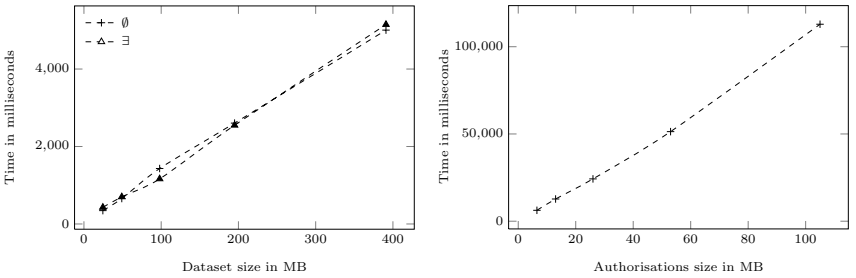
	AS_1	AS_2	AS_3	AS_4	AS_5
QS_S *query time (ms)*	5056	4801	4861	4892	4869
QS_U *query time (ms)*	9	8	9	8	9

queries; and (ii) QS_U which contained *5* INSERT and *5* DELETE queries. In both instances the queries composed of a combination of one, two and three triple patterns. Access was granted or restricted to all quads (?S ?P ?O ?G); a particular graph (?S ?P ?O G1); all quads of type offer (?S rdf:type bsbm:Offer ?G); all classes (?S rdf:type rdf:Class); and all properties (?S rdf:type rdf:Property). Users were either assigned (select; select & insert; select, insert & delete) or denied (delete; insert & delete; select, insert & delete;) access to single quad patterns. The integrity constraints presented in Examples 9 and 10 were added, to ensure that `INSERT` and `DELETE` operations were only applied to RDF quads. The conflict resolution rules presented in Examples 7 and 8 along with an additional denial takes precedence rule, were executed in the event of a conflict. The datasets, queries and the conflict resolution, integrity and propagation rules used in the experiments can be found at `http://gfaf.sabrinakirrane.com/`. All calculations presented were based on an average of 20 response times excluding the two slowest and fastest times.

In order to evaluate the enforcement algorithm we ran both the select (QS_S) and the update (QS_U) query sets, without access control ($\emptyset$), with access control for users who were granted access ($\exists$), over an authorisation set containing *588,000* grant and *402,001* deny authorisations. As expected the results indicate that select query execution times are not impacted when the dataset is increased (Table 2). However, little or no increase in performance times over increasing authorisations (Table 3 and Fig. 3a) was unexpected. Such behavior can be attributed to the fact that all authorisations are indexed by a combined `subject access right` key and subsequently by `graph pattern` (*seeAlg.* 2). For the evaluation of the propagation rules we examined the impact associated with three schema based derivations from: classes to all instances of that class; properties to all instances of that property; and an instance to property values associated with that instance. Again we ran the experiment over increasing datasets and authorisations (Table 4). Based on the results we can see that reasoning behaves

Table 4. Propagation rules performance

	DS_1	DS_2	DS_3	DS_4	DS_5
AS_5 *query time (ms)*	98531	104894	107017	106823	106248
	AS_1	AS_2	AS_3	AS_4	AS_5
DS_5 *query time (ms)*	6248	12733	24257	51339	112887



(a) Select query performance (b) Rules over increasing authorisations

Fig. 3. Query and Propagation times

linearly when the number of authorisations are increased (Fig. 3b), whereas there is little or no impact when the dataset was increased.

7 Conclusions and Future Work

With the introduction of RDF update languages, such as SPARQL 1.1, it is now feasible to both query and manage distributed and linked RDF data. However like web applications and web services, SPARQL endpoints need to protect the security of the data source and the privacy and the integrity of the data therein. In this paper, we discussed how the hierarchical Flexible Authorisation Framework, proposed by Jajodia and Samarati [9], can be adapted to cater for secure manipulation of RDF graph data. We provided a formal definition of authorisations, propagation rules, conflict resolution policies and integrity constraints, within the context of RDF, and describe how together these components can simultaneously provide access control over interlinked RDF graphs. The results of our initial performance evaluation are very promising, as in general they show only a negligible increase in query processing time and a linear increase in derivation times over increasing authorisations.

To date we have focused on the application of access control to simple graph pattern queries. In future work we will look into handling more expressive queries, for example those that include filters, subqueries, aggregates etc, and investigate integrity issues with respect to query rewriting. We also plan to extend the integrity constraints to ensure the integrity of both the rules and conflict resolution policies.

Acknowledgements. This work is supported in part by the Science Foundation Ireland under Grant No. SFI/08/CE/I1380 (Lion-2), the Irish Research Council for Science, Engineering and Technology Enterprise Partnership Scheme and Storm Technology Ltd. We would like to thank Aidan Hogan and Nuno Lopes for their valuable comments on the paper.

References

1. Abel, F., De Coi, J.L., Henze, N., Koesling, A.W., Krause, D., Olmedilla, D.: Enabling advanced and context-dependent access control in RDF stores. In: Aberer, K., Choi, K.-S., Noy, N., Allemang, D., Lee, K.-I., Nixon, L.J.B., Golbeck, J., Mika, P., Maynard, D., Mizoguchi, R., Schreiber, G., Cudré-Mauroux, P. (eds.) ASWC 2007/ISWC 2007. LNCS, vol. 4825, pp. 1–14. Springer, Heidelberg (2007)
2. Bertino, E., Samarati, P., Jajodia, S.: Authorizations in relational database management systems. In: Proceedings of the 1st ACM conference on Computer and communications security, CCS 1993, pp. 130–139 (1993)
3. Costabello, L., Villata, S., Delaforge, N.: Linked data access goes mobile: Context-aware authorization for graph stores. In: LDOW - 5th WWW Workshop on Linked Data on the Web (2012)
4. Dietzold, S., Auer, S.: Access control on RDF triple stores from a semantic wiki perspective. In: ESWC Workshop on Scripting for the Semantic Web (2006)
5. Fagin, R.: On an authorization mechanism. ACM Transactions on Database Systems (TODS) 3(3), 310–319 (1978)
6. Gabillon, A., Letouzey, L.: A View Based Access Control Model for SPARQL. In: 2010 Fourth International Conference on Network and System Security, pp. 105–112 (September 2010)
7. Griffiths, P.P., Wade, B.W.: An authorization mechanism for a relational database system. ACM Transactions on Database Systems 1, 242–255 (1976)
8. Jain, A., Farkas, C.: Secure resource description framework: an access control model. In: ACM SACMAT (2006)
9. Jajodia, S., Samarati, P.: Flexible support for multiple access control policies. ACM Trans. Database Syst. 1(212) (2001)
10. Kirrane, S., Lopes, N., Mileo, A., Decker, S.: Protect Your RDF Data! In: Proceedings of the 2nd Joint International Semantic Technology Conference (2012)
11. Kirrane, S., Mileo, A., Decker, S.: Applying DAC principles to the RDF graph data model. In: Janczewski, L.J., Wolfe, H.B., Shenoi, S. (eds.) SEC 2013. IFIP AICT, vol. 405, pp. 69–82. Springer, Heidelberg (2013)
12. Lopes, N., Kirrane, S., Zimmermann, A., Polleres, A., Mileo, A.: A Logic Programming approach for Access Control over RDF. In: Technical Communications of ICLP 2012, vol. 17, pp. 381–392. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik (2012)
13. Reddivari, P., Finin, T., Joshi, A.: Policy-Based Access Control for an RDF Store. In: Proceedings of the IJCAI 2007 Workshop on Semantic Web for Collaborative Knowledge Acquisition (January 2007)
14. Sacco, O., Passant, A., Decker, S.: An Access Control Framework for the Web of Data. In: 10th International Conference on Trust, Security and Privacy in Computing and Communications (2011)
15. Samarati, P., de Capitani di Vimercati, S.: Access control: Policies, models, and mechanisms. In: Focardi, R., Gorrieri, R. (eds.) FOSAD 2000. LNCS, vol. 2171, pp. 137–196. Springer, Heidelberg (2001)
16. Sandhu, R.S., Samarati, P.: Access control: principle and practice. IEEE Communications Magazine (1994)

A Decision Procedure for $\mathcal{SHOIQ}$ with Transitive Closure of Roles

Chan Le Duc[1], Myriam Lamolle[1], and Olivier Curé[2]

[1] LIASD Université Paris 8 - IUT de Montreuil, France
{chan.leduc,myriam.lamolle}@iut.univ-paris8.fr
[2] LIGM Université Paris-Est, France
ocure@univ-mlv.fr

Abstract. The Semantic Web makes an extensive use of the OWL DL ontology language, underlied by the $\mathcal{SHOIQ}$ description logic, to formalize its resources. In this paper, we propose a decision procedure for this logic extended with the transitive closure of roles in concept axioms, a feature needed in several application domains. The most challenging issue we have to deal with when designing such a decision procedure is to represent infinitely non-tree-shaped models, which are different from those of $\mathcal{SHOIQ}$ ontologies. To address this issue, we introduce a new blocking condition for characterizing models which may have an infinite non-tree-shaped part.

1 Introduction

The ontology language OWL-DL [1] is widely used to formalize data resources on the Semantic Web. This language is mainly based on the description logic $\mathcal{SHOIN}$ which is known to be decidable [2]. Although $\mathcal{SHOIN}$ provides *transitive roles* to model transitivity of relations, we can find several applications in which the *transitive closure of roles*, that is more expressive than transitive roles, is needed. For instance, we consider an ontology, namely $\mathcal{O}_1$, that consists of the following axioms:
$\mathsf{Human} \sqsubseteq \exists \mathsf{hasAncestor}.\{\mathsf{Eva}\}$, where $\mathsf{hasAncestor}$ is transitive
$\mathsf{hasParent} \sqsubseteq \mathsf{hasAncestor}$, $\{\mathsf{Mike}\} \sqsubseteq \mathsf{Human}$, $\{\mathsf{Mike}\} \sqsubseteq \forall \mathsf{hasParent}.\bot$

We can see that $\mathcal{O}_1$ is consistent. However, the last axiom in $\mathcal{O}_1$ would be considered as a design error which should lead to inconsistency. If the transitive role "$\mathsf{hasAncestor}$" is replaced with the transitive closure "$\mathsf{hasParent}^+$" (and the second axiom is removed), the first axiom becomes:

$$\mathsf{Human} \sqsubseteq \exists \mathsf{hasParent}^+.\{\mathsf{Eva}\}$$

It follows that the modified ontology is consistent. The point is that an instance of "$\mathsf{hasParent}^+$" represents exactly a sequence of instances of "$\mathsf{hasParent}$" while an instance of "$\mathsf{hasAncestor}$" corresponds to a sequence of instances of *itself*. In this paper, we consider an extension of $\mathcal{SHOIQ}$ by enabling transitive closure of roles in concept axioms. In the general case, transitive closure is not expressible in the first order logic [3], the logic from which DL is a sublanguage, while the second order logic is sufficiently expressive to do so.

H. Alani et al. (Eds.): ISWC 2013, Part I, LNCS 8218, pp. 264–279, 2013.

In the DL literature ([4]; [5]), there have been works dealing with transitive closure of roles. Recently, Ortiz [5] has proposed an algorithm for deciding consistency in the logic $\mathcal{ALCQI}b^{+}_{reg}$ which allows for transitive closure of roles. However, nominals are disallowed in this logic. It is known that reasoning with a DL including number restrictions, inverse roles, nominals and transitive closure of roles is hard. The reason for this is that there exists an ontology in that DL whose models have an *infinite* non-tree-shaped part. Calvanese *et al.* [6] have presented an automata-based technique for dealing with the logic $\mathcal{ZOIQ}$ that includes transitive closure of roles, and showed that the sublogics $\mathcal{ZIQ}$, $\mathcal{ZOQ}$ and $\mathcal{ZOI}$ are decidable. To obtain this result, the authors have introduced the *quasi-forest model property* to characterize models of ontologies in these sublogics. Although they are very expressive, none of these sublogics includes $\mathcal{SHOIQ}$ with transitive closure of roles, namely $\mathcal{SHOIQ}_{(+)}$. The following example[1], noted $\mathcal{K}_1$, shows that there is an ontology in $\mathcal{SHOIQ}_{(+)}$ which does not enjoy the quasi-forest model property. We consider the following axioms:

(1) $\{o\} \sqsubseteq \mathsf{A}$; $\mathsf{A} \sqcap \mathsf{B} \sqsubseteq \bot$; $\mathsf{A} \sqsubseteq \exists \mathsf{R}.\mathsf{A} \sqcap \exists \mathsf{R}'.\mathsf{B}$; $\mathsf{B} \sqsubseteq \exists \mathsf{S}^{+}.\{o\}$

(2) $\{o\} \sqsubseteq \forall \mathsf{X}^{-}.\bot$; $\top \sqsubseteq \ \leq 1\, \mathsf{X}.\top$; $\top \sqsubseteq \ \leq 1\, \mathsf{X}^{-}.\top$ where $\mathsf{X} \in \{\mathsf{R}, \mathsf{R}', \mathsf{S}\}$

Figure 1 shows an infinite non-tree-shaped model of $\mathcal{K}_1$. In fact, each individual x that satisfies $\exists \mathsf{S}^{+}.\{o\}$ must have two distinct paths from x to the individual satisfying nominal o. Intuitively, we can see that (i) such a x must satisfy $\exists \mathsf{S}^{+}.\{o\}$ and B, (ii) an individual satisfying B must connect to another individual satisfying A which must have a R-path to nominal o, and (iii) two concepts A and B are disjoint.

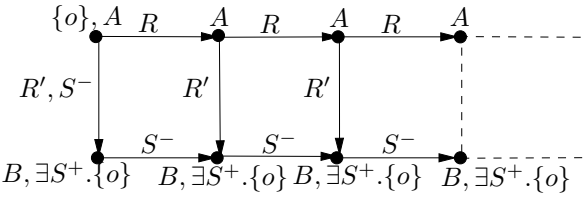


Fig. 1. An infinite non tree-shaped model of $\mathcal{K}_1$

This example shows that methods ([7], [8], [6]) based on the hypothesis which says that if an ontology is consistent it has a *quasi-forest model*, could fail to address the problem of consistency in a DL including simultaneously $\mathcal{O}$ (nominals), $\mathcal{I}$ (inverse roles), $\mathcal{Q}$ (number restrictions) and transitive closure of roles.

In this paper, we propose a decision procedure for the problem of consistency in $\mathcal{SHOIQ}$ with transitive closure of roles in concept axioms. The underlying idea of our algorithm is founded on the *star-type* and *frame* notions introduced by Pratt-Hartmann [9]. This technique uses star-types to represent individuals and "tiles" them together to form a frame for representing a model. For each star-type σ, we maintain a function $\delta(\sigma)$ which stores the number of individuals satisfying this star-type. To obtain termination, we introduce two additional structures for establishing a new blocking condition:

[1] This example is initially proposed by Sebastian Rudolph from an informal discussion.

(i) the first one, namely *cycles*, describes duplicate parts of a model resulting from interactions of logic constructors in $\mathcal{SHOIQ}$, (ii) the second one, namely *blocking-blocked cycles*, describes parts of a model bordered by cycles which allow a frame to satisfy transitive closure of roles occurring in concepts of the form $\exists R^+.C$.

2 The Description Logic $\mathcal{SHOIQ}_{(+)}$

In this section, we present the syntax, the semantics and main inference problems of $\mathcal{SHOIQ}_{(+)}$. In addition, we introduce a tableau structure for $\mathcal{SHOIQ}_{(+)}$, which allows us to represent a model of a $\mathcal{SHOIQ}_{(+)}$ knowledge base.

Definition 1. *Let* $\mathbf{R}$ *be a non-empty set of* role names *and* $\mathbf{R}_+ \subseteq \mathbf{R}$ *be a set of transitive role names. We use* $\mathbf{R}_\mathsf{I} = \{P^- \mid P \in \mathbf{R}\}$ *to denote a set of inverse roles, and* $\mathbf{R}_\oplus = \{Q^+ \mid Q \in \mathbf{R} \cup \mathbf{R}_\mathsf{I}\}$ *to denote a set of transitive closure of roles. Each element of* $\mathbf{R} \cup \mathbf{R}_\mathsf{I} \cup \mathbf{R}_\oplus$ *is called a* $\mathcal{SHOIQ}_{(+)}$*-role. A* role inclusion axiom *is of the form* $R \sqsubseteq S$ *for two* $\mathcal{SHOIQ}_{(+)}$*-roles* R *and* S *such that* $R \notin \mathbf{R}_\oplus$ *and* $S \notin \mathbf{R}_\oplus$*. A* role hierarchy $\mathcal{R}$ *is a finite set of role inclusion axioms. An interpretation* $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$ *consists of a non-empty set* $\Delta^\mathcal{I}$ *(*domain*) and a function* $\cdot^\mathcal{I}$ *which maps each role name to a subset of* $\Delta^\mathcal{I} \times \Delta^\mathcal{I}$ *such that*

$$R^{-\mathcal{I}} = \{\langle x, y\rangle \in \Delta^\mathcal{I} \times \Delta^\mathcal{I} \mid \langle y, x\rangle \in R^\mathcal{I}\} \text{ for all } R \in \mathbf{R},$$
$$\langle x, z\rangle \in S^\mathcal{I}, \langle z, y\rangle \in S^\mathcal{I} \text{ implies } \langle x, y\rangle \in S^\mathcal{I} \text{ for each } S \in \mathbf{R}_+, \text{ and}$$
$$(Q^+)^\mathcal{I} = \bigcup_{n>0} (Q^n)^\mathcal{I} \text{ with } (Q^1)^\mathcal{I} = Q^\mathcal{I},$$
$$(Q^n)^\mathcal{I} = \{\langle x, y\rangle \in (\Delta^\mathcal{I})^2 \mid \exists z \in \Delta^\mathcal{I}, \langle x, z\rangle \in (Q^{n-1})^\mathcal{I}, \langle z, y\rangle \in Q^\mathcal{I}\} \text{ for } Q^+ \in \mathbf{R}_\oplus$$

$*$ *An interpretation* $\mathcal{I}$ *satisfies a role hierarchy* $\mathcal{R}$ *if* $R^\mathcal{I} \subseteq S^\mathcal{I}$ *for each* $R \sqsubseteq S \in \mathcal{R}$*. Such an interpretation is called a* model *of* $\mathcal{R}$*, denoted by* $\mathcal{I} \models \mathcal{R}$*. To simplify notations for nested inverse roles and transitive closures of roles, we define two functions* $\cdot^\ominus$ *and* $\cdot^\oplus$ *as follows:*

$$R^\ominus = \begin{cases} R^- & \text{if } R \in \mathbf{R}; \\ S & \text{if } R = S^- \text{ and } S \in \mathbf{R}; \\ (S^-)^+ & \text{if } R = S^+, S \in \mathbf{R}, \\ S^+ & \text{if } R = (S^-)^+, S \in \mathbf{R} \end{cases} \quad R^\oplus = \begin{cases} R^+ & \text{if } R \in \mathbf{R}; \\ S^+ & \text{if } R = (S^+)^+ \text{ and } S \in \mathbf{R}; \\ (S^-)^+ & \text{if } R = S^- \text{ and } S \in \mathbf{R}; \\ (S^-)^+ & \text{if } R = (S^+)^- \text{ and } S \in \mathbf{R} \end{cases}$$

$*$ *A relation* $\boxtimes$ *is defined as the transitive-reflexive closure* $\mathcal{R}^+$ *of* $\sqsubseteq$ *on* $\mathcal{R} \cup \{R^\ominus \sqsubseteq S^\ominus \mid R \sqsubseteq S \in \mathcal{R}\} \cup \{R^\oplus \sqsubseteq S^\oplus \mid R \sqsubseteq S \in \mathcal{R}\} \cup \{Q \sqsubseteq Q^\oplus \mid Q \in \mathbf{R} \cup \mathbf{R}_\mathsf{I}\}$*. We define a function* $\mathsf{Trans}(R)$ *which returns* true *iff there is some* $Q \in \mathbf{R}_+ \cup \{P^\ominus \mid P \in \mathbf{R}_+\} \cup \{P^\oplus \mid P \in \mathbf{R} \cup \mathbf{R}_\mathsf{I}\}$ *such that* $Q \boxtimes R \in \mathcal{R}^+$*. A role* R *is called* simple *w.r.t.* $\mathcal{R}$ *if* $\mathsf{Trans}(R) = \mathsf{false}$.

The reason for the introduction of two functions $\cdot^\ominus$ and $\cdot^\oplus$ in Definition 1 is that they avoid using R^{--} and R^{++}. Moreover, it remains a unique nested case $(R^-)^+$. According to Definition 1, axiom $R \sqsubseteq Q^\oplus$ is not allowed in a role hierarchy $\mathcal{R}$ since this may lead to undecidability [10] even if R is simple. Notice that the closure $\mathcal{R}^+$ may contain $R \sqsubseteq Q^\oplus$ if $R \sqsubseteq Q$ belongs to $\mathcal{R}$.

Definition 2 (terminology). *Let* $\mathbf{C}$ *be a non-empty set of* concept names *with a non-empty subset* $\mathbf{C}_o \subseteq \mathbf{C}$ *of nominals. The set of* $\mathcal{SHOIQ}_{(+)}$*-concepts is inductively defined as the smallest set containing all* C *in* $\mathbf{C}$*,* $\top$*,* $C \sqcap D$*,* $C \sqcup D$*,* $\neg C$*,* $\exists R.C$*,* $\forall R.C$*,* $(\leq n\, S.C)$ *and* $(\geq n\, S.C)$ *where* n *is a positive integer,* C *and* D *are* $\mathcal{SHOIQ}_{(+)}$*-concepts,* R *is an* $\mathcal{SHOIQ}_{(+)}$*-role and* S *is a simple role w.r.t. a role hierarchy. We denote* $\bot$ *for* $\neg\top$*. The interpretation function* $\cdot^{\mathcal{I}}$ *of an interpretation* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ *maps each concept name to a subset of* $\Delta^{\mathcal{I}}$ *such that* $\top^{\mathcal{I}} = \Delta^{\mathcal{I}}$*,* $(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$*,* $(C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}}$*,* $(\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \backslash C^{\mathcal{I}}$*,* $|\{o^{\mathcal{I}}\}| = 1$ *for all* $o \in \mathbf{C}_o$*,* $(\exists R.C)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid \exists y \in \Delta^{\mathcal{I}}, \langle x, y\rangle \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$*,* $(\forall R.C)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid \forall y \in \Delta^{\mathcal{I}}, \langle x, y\rangle \in R^{\mathcal{I}} \Rightarrow y \in C^{\mathcal{I}}\}$*,* $(\geq n\, S.C)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid |\{y \in C^{\mathcal{I}} \mid \langle x, y\rangle \in S^{\mathcal{I}}| \geq n\}$*,* $(\leq n\, S.C)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid |\{y \in C^{\mathcal{I}} \mid \langle x, y\rangle \in S^{\mathcal{I}}| \leq n\}$ *where* $|S|$ *is denoted for the cardinality of a set* S*. An axiom* $C \sqsubseteq D$ *is called a general concept inclusion (GCI) where* C, D *are* $\mathcal{SHOIQ}_{(+)}$*-concepts (possibly complex), and a finite set of GCIs is called a terminology* $\mathcal{T}$*. An interpretation* $\mathcal{I}$ *satisfies a GCI* $C \sqsubseteq D$ *if* $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ *and* $\mathcal{I}$ *satisfies a terminology* $\mathcal{T}$ *if* $\mathcal{I}$ *satisfies each GCI in* $\mathcal{T}$*. Such an interpretation is called a* model *of* $\mathcal{T}$*, denoted by* $\mathcal{I} \models \mathcal{T}$*. A pair* $(\mathcal{T}, \mathcal{R})$ *is called a* $\mathcal{SHOIQ}_{(+)}$ *knowledge base where* $\mathcal{R}$ *is a* $\mathcal{SHOIQ}_{(+)}$ *role hierarchy and* $\mathcal{T}$ *is a* $\mathcal{SHOIQ}_{(+)}$ *terminology. A knowledge base* $(\mathcal{T}, \mathcal{R})$ *is said to be consistent if there is a model* $\mathcal{I}$ *of both* $\mathcal{T}$ *and* $\mathcal{R}$*, i.e.,* $\mathcal{I} \models \mathcal{T}$ *and* $\mathcal{I} \models \mathcal{R}$*. A concept* C *is called* satisfiable *w.r.t.* $(\mathcal{T}, \mathcal{R})$ *iff there is some interpretation* $\mathcal{I}$ *such that* $\mathcal{I} \models \mathcal{R}$*,* $\mathcal{I} \models \mathcal{T}$ *and* $C^{\mathcal{I}} \neq \emptyset$*. Such an interpretation is called a* model *of* C *w.r.t.* $(\mathcal{T}, \mathcal{R})$*. A concept* D subsumes *a concept* C *w.r.t.* $(\mathcal{T}, \mathcal{R})$*, denoted by* $C \sqsubseteq D$*, if* $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ *holds in each model* $\mathcal{I}$ *of* $(\mathcal{T}, \mathcal{R})$*.* $\lhd$

Since unsatisfiability, subsumption and consistency w.r.t. a $\mathcal{SHOIQ}_{(+)}$ knowledge base can be reduced to each other, it suffices to study knowledge base consistency. For the ease of construction, we assume all concepts to be in *negation normal form* (NNF), i.e., negation occurs only in front of concept names. Any $\mathcal{SHOIQ}_{(+)}$-concept can be transformed to an equivalent one in NNF by using DeMorgan's laws and some equivalences as presented in [11]. According to [12], $\mathsf{nnf}(C)$ can be computed in polynomial time in the size of C. For a concept C, we denote the nnf of C by $\mathsf{nnf}(C)$ and the nnf of $\neg C$ by $\dot{\neg} C$. Let D be a $\mathcal{SHOIQ}_{(+)}$-concept in NNF. We define $\mathsf{cl}(D)$ to be the smallest set that contains all sub-concepts of D including D. For a knowledge base $(\mathcal{T}, \mathcal{R})$, we reuse $\mathsf{cl}(\mathcal{T}, \mathcal{R})$ introduced by Horrocks *et al.* [7] to denote all sub-concepts occurring in the axioms of $(\mathcal{T}, \mathcal{R})$ as follows:

$$\mathsf{cl}(\mathcal{T}, \mathcal{R}) = \bigcup_{C \sqsubseteq D \in \mathcal{T}} \mathsf{cl}(\mathsf{nnf}(\neg C \sqcup D), \mathcal{R}) \ \textit{where}$$

$$\mathsf{cl}(E, \mathcal{R}) = \mathsf{cl}(E) \cup \{\dot{\neg} C \mid C \in \mathsf{cl}(E)\} \ \cup \tag{1}$$

$$\{\forall S.C \mid (\forall R.C \in \mathsf{cl}(E), S \sqsubseteq R) \textit{ or } (\dot{\neg}\forall R.C \in \mathsf{cl}(E), S \sqsubseteq R) \textit{ where } S \textit{ occurs in } \mathcal{T} \textit{ or } \mathcal{R}\} \ \cup \tag{2}$$

$$\bigcup_{\exists Q^{\oplus}.C \textit{ occurs in } \mathcal{T}} \mathsf{cl}(\exists Q.C \sqcup \exists Q.\exists Q^{\oplus}.C) \tag{3}$$

Since (1) consists of sub-concepts from $\mathcal{T}$ and (2) is formed from concepts in (1) by replacing a role or a logic constructor with respective another role occurring in $\mathcal{R}$ or

another logic constructor, both of these sets are bounded by $\mathcal{O}(|(\mathcal{T},\mathcal{R})|)$. Thus, $\mathsf{cl}(\mathcal{T},\mathcal{R})$ is bounded by $\mathcal{O}(|(\mathcal{T},\mathcal{R})|)$.

We have $\mathsf{cl}(\mathcal{T},\mathcal{R})$ is bounded by $\mathcal{O}(|(\mathcal{T},\mathcal{R})|)$ [7]. To translate *star-type* and *frame* structures presented by Pratt-Hartmann (2005) for $\mathcal{C}^2$ into those for $\mathcal{SHOIQ}$, we need to add new sets of concepts, denoted $\mathsf{cl}_1(\mathcal{T},\mathcal{R})$ and $\mathsf{cl}_2(\mathcal{T},\mathcal{R})$, to the signature of a $\mathcal{SHOIQ}_{(+)}$ knowledge base $(\mathcal{T},\mathcal{R})$.

$$\mathsf{cl}_1(\mathcal{T},\mathcal{R}) = \{\leq mS.C \mid \{(\leq nS.C),(\geq nS.C)\} \cap \mathsf{cl}(\mathcal{T},\mathcal{R}) \neq \emptyset, 1 \leq m \leq n\} \cup \{\geq mS.C \mid \{(\leq nS.C),(\geq nS.C)\} \cap \mathsf{cl}(\mathcal{T},\mathcal{R}) \neq \emptyset, 1 \leq m \leq n\}$$

For a generating concept $(\geq nS.C)$ and a set $I \subseteq \{0,\cdots,\lceil log\ n+1\rceil\}$, we denote $\mathscr{C}^I_{(\geq nS.C)} = \prod_{i\in I} C^i_{(\geq nS.C)} \sqcap \prod_{j\notin I} \neg C^j_{(\geq nS.C)}$ where $C^i_{(\geq nS.C)}$ are new concept names for $0 \leq i \leq \lceil log\ n+1\rceil$. We define $\mathsf{cl}_2(\mathcal{T},\mathcal{R})$ as follows:

$$\mathsf{cl}_2(\mathcal{T},\mathcal{R}) = \{C^i_{(\geq S.C)} \mid (\geq nS.C) \in \mathsf{cl}(\mathcal{T},\mathcal{R}) \cup \mathsf{cl}_1(\mathcal{T},\mathcal{R}), 0 \leq i \leq \lceil log\ n+1\rceil\} \cup \{\mathscr{C}^I_{(\geq nS.C)} \mid (\geq nS.C) \in \mathsf{cl}(\mathcal{T},\mathcal{R}) \cup \mathsf{cl}_1(\mathcal{T},\mathcal{R}), I \subseteq \{0,\cdots,\lceil log\ n+1\rceil\}\}$$

Remark 1. If numbers are encoded in binary then the number of new concept names $C^i_{(\geq nS.D)}$ for $0 \leq i \leq \lceil log\ n+1\rceil$, is bounded by $\mathcal{O}(|(\mathcal{T},\mathcal{R})|)$ since n is bounded by $\mathcal{O}(2^{|(\mathcal{T},\mathcal{R})|})$. This implies that $|\mathsf{cl}_2(\mathcal{T},\mathcal{R})|$ is bounded by $\mathcal{O}(|(\mathcal{T},\mathcal{R})|)$. Note that two concepts $\mathscr{C}^I_{(\geq nS.C)}$ and $\mathscr{C}^J_{(\geq nS.C)}$ are disjoint for all $I, J \subseteq \{0,\cdots,\lceil log\ n+1\rceil\}$, $I \neq J$. The concepts $\mathscr{C}_{(\exists S.C)}$ and $\mathscr{C}^I_{(\geq nS.C)}$ will be used for building chromatic star-types. This notion will be clarified after introducing the frame structure (Definition 6).

Finally, we denote $\mathbf{CL}(\mathcal{T},\mathcal{R}) = \mathsf{cl}(\mathcal{T},\mathcal{R}) \cup \mathsf{cl}_1(\mathcal{T},\mathcal{R}) \cup \mathsf{cl}_2(\mathcal{T},\mathcal{R})$, and use $\mathbf{R}(\mathcal{T},\mathcal{R})$ to denote the set of all role names occurring in $\mathcal{T},\mathcal{R}$ with their inverse. The definition of $\mathbf{CL}(\mathcal{T},\mathcal{R})$ is inspired from the Fischer-Ladner closure that was introduced in [13]. The closure $\mathbf{CL}(\mathcal{T},\mathcal{R})$ contains not only sub-concepts syntactically obtained from $\mathcal{T}$ but also sub-concepts that are semantically derived from $\mathcal{T}$ w.r.t. $\mathcal{R}$. For instance, if $\forall S.C$ is a sub-concept from $\mathcal{T}$ and $R \sqsubseteq S \in \mathcal{R}$ then $\forall R.C \in \mathbf{CL}(\mathcal{T},\mathcal{R})$.

To describe a model of a $\mathcal{SHOIQ}_{(+)}$ knowledge base in a more intuitive way, we use a tableau structure that expresses semantic constraints resulting directly from the logic constructors in $\mathcal{SHOIQ}_{(+)}$.

Definition 3. *Let $(\mathcal{T},\mathcal{R})$ be an $\mathcal{SHOIQ}_{(+)}$ knowledge base. A tableau T for $(\mathcal{T},\mathcal{R})$ is defined to be a triplet $(\mathbf{S},\mathcal{L},\mathcal{E})$ such that $\mathbf{S}$ is a set of individuals, $\mathcal{L}$: $\mathbf{S} \to 2^{\mathbf{CL}(\mathcal{T},\mathcal{R})}$ and $\mathcal{E}$: $\mathbf{R}(\mathcal{T},\mathcal{R}) \to 2^{\mathbf{S}\times\mathbf{S}}$. For all $s,t \in \mathbf{S}$, $C, C_1, C_2 \in \mathbf{CL}(\mathcal{T},\mathcal{R})$, and $R, S, Q^{\oplus} \in \mathbf{R}(\mathcal{T},\mathcal{R})$, T satisfies the following properties:*

- **P1** *If $C_1 \sqsubseteq C_2 \in \mathcal{T}$ and $s \in \mathbf{S}$ then $\mathsf{nnf}(\neg C_1 \sqcup C_2) \in \mathcal{L}(s)$;*
- **P2** *If $C \in \mathcal{L}(s)$, then $\neg C \notin \mathcal{L}(s)$;*
- **P3** *If $C_1 \sqcap C_2 \in \mathcal{L}(s)$, then $C_1 \in \mathcal{L}(s)$ and $C_2 \in \mathcal{L}(s)$;*
- **P4** *If $C_1 \sqcup C_2 \in \mathcal{L}(s)$, then $C_1 \in \mathcal{L}(s)$ or $C_2 \in \mathcal{L}(s)$;*
- **P5** *If $\forall S.C \in \mathcal{L}(s)$ and $\langle s,t\rangle \in \mathcal{E}(S)$, then $C \in \mathcal{L}(t)$;*
- **P6** *If $\exists S.C \in \mathcal{L}(s)$ then there is some $t \in \mathbf{S}$ such that $\langle s,t\rangle \in \mathcal{E}(S)$ and $\{C, \mathscr{C}_{(\exists S.C)}\} \subseteq \mathcal{L}(t)$;*
- **P7** *If $\forall S.C \in \mathcal{L}(s)$ and $\langle s,t\rangle \in \mathcal{E}(R)$ for $R \sqsubseteq S$ and $\mathsf{Trans}(R)$ then $\forall R.C \in \mathcal{L}(t)$;*

P8 *If* $\exists Q^{\oplus}.C \in \mathcal{L}(s)$ *then* $(\exists Q.C \sqcup \exists Q.\exists Q^{\oplus}.C) \in \mathcal{L}(s)$ *and there are* $s_1, \cdots, s_{n-1} \in \mathbf{S}$ *such that* $\exists Q.C \in \mathcal{L}(s_0) \cup \mathcal{L}(s_{n-1})$, $\langle s_i, s_{i+1} \rangle \in \mathcal{E}(Q)$ *with* $0 \leq i < n-1$, $s_0 = s$ *and* $\exists Q^{\oplus}.C \in \mathcal{L}(s_j)$ *for all* $0 \leq j < n-1$.
P9 $\langle s, t \rangle \in \mathcal{E}(R)$ *iff* $\langle t, s \rangle \in \mathcal{E}(R^{\ominus})$;
P10 *If* $\langle s, t \rangle \in \mathcal{E}(R)$, $R \sqsubseteq S$ *then* $\langle s, t \rangle \in \mathcal{E}(S)$;
P11 *If* $(\geq n\ S\ C) \in \mathcal{L}(s)$ *then there are* $t_1, \cdots, t_n \in \mathbf{S}$ *such that* $\{C, \mathscr{C}^{I_i}_{(\geq nS.C)}\} \subseteq \mathcal{L}(t_i)$ *and* $\langle s, t_i \rangle \in \mathcal{E}(S)$ *for all* $1 \leq i \leq n$, *and* $I_j, I_k \subseteq \{0, \cdots, \lceil \log n + 1 \rceil\}$, $I_j \neq I_k$ *for all* $1 \leq j < k \leq n$;
P12 *If* $(\leq n\ S\ C) \in \mathcal{L}(s)$ *then* $|S^T(s, C)| \leq n$;
P13 *If* $(\leq n\ S\ C) \in \mathcal{L}(s)$ *and* $\langle s, t \rangle \in \mathcal{E}(S)$ *then* $\{C, \neg C\} \cap \mathcal{L}(t) \neq \emptyset$ *where* $S^T(s, C) := \{t \in \mathbf{S} | \langle s, t \rangle \in \mathcal{E}(S) \wedge C \in \mathcal{L}(t)\}$;
P14 *If* $o \in \mathcal{L}(s) \cap \mathcal{L}(t)$ *for some* $o \in \mathbf{C}_o$ *then* $s = t$.
P15 *For each* $o \in \mathbf{C}_o$, *if* o *occurs in* $\mathcal{T}$ *then there is* $s \in \mathbf{S}$ *such that* $o \in \mathcal{L}(s)$.

Note that the property P8 is added to deal with transitive closure of roles. The following lemma establishes the equivalence between a model of an ontology and a tableau.

Lemma 1. *Let* $(\mathcal{T}, \mathcal{R})$ *be a* $\mathcal{SHOIQ}_{(+)}$ *knowledge base.* $(\mathcal{T}, \mathcal{R})$ *is consistent iff there is a tableau for* $(\mathcal{T}, \mathcal{R})$.

A proof of Lemma 1 can be found in [14].

3 A Decision Procedure For $\mathcal{SHOIQ}_{(+)}$

This section starts by translating *star-type* and *frame* structures presented by Pratt-Hartmann (2005) for $\mathcal{C}^2$ into those for $\mathcal{SHOIQ}_{(+)}$.

Definition 4 (star-type). *Let* $(\mathcal{T}, \mathcal{R})$ *be a* $\mathcal{SHOIQ}_{(+)}$ *knowledge base. A* star-type *is a pair* $\sigma = \langle \lambda(\sigma), \xi(\sigma) \rangle$, *where* $\lambda(\sigma) \in 2^{\mathbf{CL}(\mathcal{T}, \mathcal{R})}$ *is called* core label, $\xi(\sigma) = (\langle r_1, l_1 \rangle, \cdots, \langle r_d, l_d \rangle)$ *is a* d*-tuple over* $2^{\mathbf{R}(\mathcal{T}, \mathcal{R})} \times 2^{\mathbf{CL}(\mathcal{T}, \mathcal{R})}$. *A pair* $\langle r, l \rangle$ *is a* ray *of* σ *if* $\langle r, l \rangle = \langle r_i, l_i \rangle$ *for some* $1 \leq i \leq d$. *We use* $\langle r(\rho), l(\rho) \rangle$ *to denote a ray* $\rho = \langle r, l \rangle$ *where* $r(\rho) = r$ *and* $l(\rho) = l$.

- *A star-type* σ *is* nominal *if* $o \in \lambda(\sigma)$ *for some* $o \in \mathbf{C}_o$.
- *A star-type* σ *is* chromatic *if* $\rho \neq \rho'$ *implies* $l(\rho) \neq l(\rho')$ *for two rays* ρ, ρ' *of* σ. *When a star-type* σ *is chromatic,* $\xi(\sigma)$ *can be considered as a set of rays.*
- *Two star-types* σ, σ' *are* equivalent *if* $\lambda(\sigma) = \lambda(\sigma')$, *and there is a bijection* π *between* $\xi(\sigma)$ *and* $\xi(\sigma')$ *such that* $\pi(\rho) = \rho'$ *implies* $r(\rho') = r(\rho)$ *and* $l(\rho') = l(\rho)$.

We denote Σ *for the set of all star-types for* $(\mathcal{T}, \mathcal{R})$. ◁

Note that for a chromatic star-type σ, $\xi(\sigma)$ can be considered as a set of rays since rays are distinct and not ordered. We can think of a star-type σ as the set of individuals x satisfying all concepts in $\lambda(\sigma)$, and each ray ρ of σ corresponds to a "neighbor" individual x_i of x such that $r(\rho)$ is the label of the link between x and x_i; and x_i satisfies all concepts in $l(\rho)$. In this case, we say that x *satisfies* σ.

Definition 5 (valid star-type). *Let $(\mathcal{T}, \mathcal{R})$ be a $\mathcal{SHOIQ}_{(+)}$ knowledge base. Let σ be a star-type for $(\mathcal{T}, \mathcal{R})$ where $\sigma = \langle \lambda(\sigma), \xi(\sigma) \rangle$. The star-type σ is* valid *if σ is chromatic and the following conditions are satisfied:*

1. *If $C \sqsubseteq D \in \mathcal{T}$ then $\mathsf{nnf}(\neg C \sqcup D) \in \lambda(\sigma)$;*
2. *$\{A, \neg A\} \not\subseteq \lambda$ for every concept name A where $\lambda = \lambda(\sigma)$ or $\lambda = l(\rho)$ for each $\rho \in \xi(\sigma)$;*
3. *If $C_1 \sqcap C_2 \in \lambda(\sigma)$ then $\{C_1, C_2\} \subseteq \lambda(\sigma)$;*
4. *If $C_1 \sqcup C_2 \in \lambda(\sigma)$ then $\{C_1, C_2\} \cap \lambda(\sigma) \neq \emptyset$;*
5. *If $\exists R.C \in \lambda(\sigma)$ then there is some ray $\rho \in \xi(\sigma)$ such that $C \in l(\rho)$ and $R \in r(\rho)$;*
6. *If $(\leq nS.C) \in \lambda(\sigma)$ and there is some ray $\rho \in \xi(\sigma)$ such that $S \in r(\rho)$ then $C \in l(\rho)$ or $\dot{\neg} C \in l(\rho)$;*
7. *If $(\leq nS.C) \in \lambda(\sigma)$ and there is some ray $\rho \in \xi(\sigma)$ such that $C \in l(\rho)$ and $S \in r(\rho)$ then there is some $1 \leq m \leq n$ such that $\{(\leq mS.C), (\geq mS.C)\} \subseteq \lambda(\sigma)$;*
8. *For each ray $\rho \in \xi(\sigma)$, if $R \in r(\rho)$ and $R \trianglerighteq S$ then $S \in r(\rho)$;*
9. *If $\forall R.C \in \lambda(\sigma)$ and $R \in r(\rho)$ for some ray $\rho \in \xi(\sigma)$ then $C \in l(\rho)$;*
10. *If $\forall R.D \in \lambda(\sigma)$, $S \trianglerighteq R$, $\mathsf{Trans}(S)$ and $R \in r(\rho)$ for some ray $\rho \in \xi(\sigma)$ then $\forall S.D \in l(\rho)$;*
11. *If $\exists Q^{\oplus}.C \in \lambda(\sigma)$ then $(\exists Q.C \sqcup \exists Q.\exists Q^{\oplus}.C) \in \lambda(\sigma)$;*
12. *If $(\geq nS.C) \in \lambda(\sigma)$ then there are n distinct rays $\rho_1, \cdots, \rho_n \in \xi(\sigma)$ such that $\{C, \mathscr{C}^{I_i}_{(\geq nS.C)}\} \subseteq l(\rho_i)$, $S \in r(\rho_i)$ for all $1 \leq i \leq n$; and $I_j, I_k \subseteq \{0, \cdots, log\, n + 1\}$, $I_j \neq I_k$ for all $1 \leq j < k \leq n$;*
13. *If $(\leq nS.C) \in \lambda(\sigma)$ and there do not exist $n+1$ rays $\rho_0, \cdots, \rho_n \in \xi(\sigma)$ such that $C \in l(\rho_i)$ and $S \in r(\rho_i)$ for all $0 \leq i \leq n$.* ◁

Roughly speaking, a star-type σ is valid if each individual x satisfies *semantically* all concepts in $\lambda(\sigma)$. In fact, each condition in Definition 5 represents the semantics of a constructor in $\mathcal{SHOIQ}_{(+)}$ except for transitive closure of roles. From valid star-types, we can "tile" a model instead of using expansion rules for generating nodes as described in tableau algorithms. Before presenting how to "tile" a model from star-types, we need some notation that will be used in the remainder of the paper.

Notation 1. *We call $\mathcal{P} = \langle (\sigma_1, \rho_1, d_1), \cdots, (\sigma_k, \rho_k, d_k) \rangle$ a* sequence *where $\sigma_i \in \Sigma$, $\rho_i \in \xi(\sigma_i)$ and $d_i \in \mathbb{N}$ for $1 \leq i \leq k$.*

- $\mathsf{tail}(\mathcal{P}) = (\sigma_k, \rho_k, d_k)$, $\mathsf{tail}_\sigma(\mathcal{P}) = \sigma_k$, $\mathsf{tail}_\rho(\mathcal{P}) = \rho_k$, $\mathsf{tail}_\delta(\mathcal{P}) = d_k$ *and* $|\mathcal{P}| = k$. *We denote* $\mathcal{L}(\mathcal{P}) = \lambda(\mathsf{tail}_\sigma(\mathcal{P}))$.
- $\mathsf{p}^i(\mathcal{P}) = (\sigma_i, \rho_i, d_i)$, $\mathsf{p}^i_\sigma(\mathcal{P}) = \sigma_i$, $\mathsf{p}^i_\rho(\mathcal{P}) = \rho_i$ *and* $\mathsf{p}^i_\delta(\mathcal{P}) = d_i$ *for each* $1 \leq i \leq k$.
- *an operation* $\mathsf{add}(\mathcal{P}, (\sigma, \rho, d))$ *extends $\mathcal{P}$ to a new sequence with* $\mathsf{add}(\mathcal{P}, (\sigma, \rho, d)) = \langle \mathcal{P}, (\sigma, \rho, d) \rangle$.

Definition 6 (frame). *Let $(\mathcal{T}, \mathcal{R})$ be a $\mathcal{SHOIQ}_{(+)}$ knowledge base. A* frame *for $(\mathcal{T}, \mathcal{R})$ is a tuple $\mathcal{F} = \langle \mathcal{N}, \mathcal{N}_o, \Omega, \delta \rangle$, where*

1. *$\mathcal{N}$ is a set of valid star-types such that σ is not equivalent to σ' for all $\sigma, \sigma' \in \mathcal{N}$;*
2. *$\mathcal{N}_o \subseteq \mathcal{N}$ is a set of nominal star-types;*

3. *Ω is a function that maps each pair (σ, ρ) with $\sigma \in \mathcal{N}$ and $\rho \in \xi(\sigma)$ to a sequence $\Omega(\sigma, \rho) = \langle(\sigma_1, \rho_1, d_1), \cdots, (\sigma_m, \rho_m, d_m)\rangle$ with $\sigma_i \in \mathcal{N}$, $\rho_i \in \xi(\sigma_i)$, $d_i \in \mathbb{N}$ for $1 \leq i \leq m$ such that for each σ_i with $1 \leq i \leq m$, it holds that $l(\rho) = \lambda(\sigma_i)$, $l(\rho_i) = \lambda(\sigma)$ and $r(\rho_i) = r^-(\rho)$ where $r^-(\rho) = \{R^\ominus \mid R \in r(\rho)\}$.*
4. *δ is a function $\delta : \mathcal{N} \to \mathbb{N}$. By abuse of notation, we also use δ to denote a function which maps each pair (σ, ρ) with $\sigma \in \mathcal{N}$ and $\rho \in \xi(\sigma)$ into a number in $\mathbb{N}$, i.e., $\delta(\sigma, \rho) \in \mathbb{N}$.* $\lhd$

Since a frame cannot contain two equivalent star-type (Condition 1 in Definition 6), the number of different star-types in a frame is bounded. The following lemma provides such a bound.

Lemma 2. *Let $\mathcal{F} = \langle \mathcal{N}, \mathcal{N}_o, \Omega, \delta\rangle$ be a frame for a $\mathcal{SHOIQ}_{(+)}$ knowledge base $(\mathcal{T}, \mathcal{R})$. The number of different star-types is bounded by $\mathcal{O}(2^{2^{|(\mathcal{T},\mathcal{R})|}})$.*

The lemma is a consequence of the following facts : (i) the number of different core labels of star-types is bounded by $\mathcal{O}(|(\mathcal{T}, \mathcal{R})|)$, (ii) the number of different ray labels of star-types is bounded by $\mathcal{O}(2^{|(\mathcal{T},\mathcal{R})|})$, and (iii) the number of different rays of a star-type is bounded by $\mathcal{O}(2^{|(\mathcal{T},\mathcal{R})|})$ due to binary coding of numbers.

The frame structure, as introduced in Definition 6, allows us to compress individuals of a model into star-types. For each star-type σ and each ray $\rho \in \xi(\sigma)$, a list $\Omega(\sigma, \rho)$ of triples (σ_i, ρ_i, d_i) with $\rho_i \in \xi(\sigma_i)$ is maintained where σ_i is a "neighbor" star-type of σ via $\rho \in \xi(\sigma)$, and d_i indicates the d_i-th "layer" of rays of σ_i. We can think a layer of rays of σ_i as an individual that connects to its neighbor individuals via the rays of σ_i. The following definition presents how to connect such layers to form paths in a frame.

Definition 7 (path). *Let $\mathcal{F} = \langle \mathcal{N}, \mathcal{N}_o, \Omega, \delta\rangle$ be a frame for a $\mathcal{SHOIQ}_{(+)}$ knowledge base $(\mathcal{T}, \mathcal{R})$. A* path *is inductively defined as follows:*

1. *A sequence $\langle\emptyset, (\sigma, \rho, 1)\rangle$ is a path, namely* nominal path*, if $\sigma \in \mathcal{N}_o$ and $\rho \in \xi(\sigma)$;*
2. *A sequence $\langle\mathcal{P}, (\sigma, \rho, d)\rangle$ with $\mathcal{P} \neq \emptyset$ and $\mathsf{tail}(\mathcal{P}) = (\sigma_0, \rho_0, d_0)$, is a path if $(\sigma, \rho, d) = \mathsf{p}^{d_0}(\Omega(\sigma_0, \rho'))$ for each $\rho' \neq \rho_0$. In this case, we say that $\langle\mathcal{P}, (\sigma, \rho, d)\rangle$ is the ρ'-*neighbor *of $\mathcal{P}$, and two paths $\mathcal{P}$, $\langle\mathcal{P}, (\sigma, \rho, d)\rangle$ are neighbors. Additionally, if $\langle\mathcal{P}, (\sigma, \rho, d)\rangle$ is a ρ'-neighbor of $\mathcal{P}$ and $Q \in r(\rho')$ then $\langle\mathcal{P}, (\sigma, \rho, d)\rangle$ is a Q-neighbor of $\mathcal{P}$. In this case, we say that $\langle\mathcal{P}, (\sigma, \rho, d)\rangle$ is a Q-neighbor of $\mathcal{P}$, or $\mathcal{P}$ is a $Q^\ominus$-neighbor of $\langle\mathcal{P}, (\sigma, \rho, d)\rangle$.*

We define $\mathcal{P} \sim \mathcal{P}'$ if $\mathsf{tail}_\sigma(\mathcal{P}) = \mathsf{tail}_\sigma(\mathcal{P}')$ and $\mathsf{tail}_\delta(\mathcal{P}) = \mathsf{tail}_\delta(\mathcal{P}')$. Since $\sim$ is an equivalence relation over the set of all paths, we use $\mathscr{P}$ to denote the set of all equivalence classes $[\mathcal{P}]$ of paths in $\mathcal{F}$. For $[\mathcal{P}], [\mathcal{Q}] \in \mathscr{P}$, we define:

1. *$[\mathcal{P}]$ is a neighbor (ρ'-neighbor) of $[\mathcal{Q}]$ if there are $\mathcal{P}' \in [\mathcal{P}]$ and $\mathcal{Q}' \in [\mathcal{Q}]$ such that $\mathcal{Q}'$ is a neighbor (ρ'-neighbor) of $\mathcal{P}'$;*
2. *$[\mathcal{Q}]$ is a reachable path of $[\mathcal{P}]$ via a ray $\rho \in \xi(\mathsf{tail}_\sigma(\mathcal{P}))$ if there are $[\mathcal{P}_1], \cdots, [\mathcal{P}_n] \in \mathscr{P}$ such that $[\mathcal{P}_i] \neq [\mathcal{P}_j]$ for $1 \leq i < j \leq n$, $[\mathcal{P}] = [\mathcal{P}_1]$, $[\mathcal{Q}] = [\mathcal{P}_n]$, $[\mathcal{P}_2]$ is the ρ-neighbor of $[\mathcal{P}_1]$, $[\mathcal{P}_{i+1}]$ is a neighbor of $[\mathcal{P}_i]$ for all $1 \leq i < n - 1$.*
3. *$[\mathcal{Q}]$ is a Q-neighbor of $[\mathcal{P}]$ if there are $\mathcal{P}' \in [\mathcal{P}]$ and $\mathcal{Q}' \in [\mathcal{Q}]$ such that $\mathcal{Q}'$ is a Q-neighbor of $\mathcal{P}'$, or $\mathcal{P}'$ is a $Q^\ominus$-neighbor of $\mathcal{Q}'$;*

4. *$[\mathcal{Q}]$ is a Q-reachable path of $[\mathcal{P}]$ if there are $[\mathcal{P}_1], \cdots, [\mathcal{P}_n] \in \mathscr{P}$ such that $[\mathcal{P}_i] \neq [\mathcal{P}_j]$ for $1 \leq i < j \leq n$, $[\mathcal{P}] = [\mathcal{P}_1]$, $[\mathcal{Q}] = [\mathcal{P}_n]$, $[\mathcal{P}_2]$ is the ρ-neighbor of $[\mathcal{P}_1]$, and $[\mathcal{P}_{i+1}]$ is a Q-neighbor of $[\mathcal{P}_i]$ for all $1 \leq i < n$.* ◁

Since two paths $\mathcal{P}$ and $\mathcal{P}'$ meet at the same star-type (i.e. $\mathsf{tail}_\sigma(\mathcal{P}) = \mathsf{tail}_\sigma(\mathcal{P}')$) and the same layer (i.e. $\mathsf{tail}_\delta(\mathcal{P}) = \mathsf{tail}_\delta(\mathcal{P}')$) should be considered as identical, we define the equivalence relation $\sim$ in Definition 7 to formalize this idea. Note that for two paths $\mathcal{P}, \mathcal{P}'$ with $\mathsf{tail}_\rho(\mathcal{P}) \neq \mathsf{tail}_\rho(\mathcal{P}')$, we have $\mathcal{P} \sim \mathcal{P}'$ if $\mathsf{tail}_\sigma(\mathcal{P}) = \mathsf{tail}_\sigma(\mathcal{P}')$ and $\mathsf{tail}_\delta(\mathcal{P}) = \mathsf{tail}_\delta(\mathcal{P}')$. This does not allow for extending $\mathsf{tail}_\rho(\mathcal{P})$ to $\mathsf{tail}_\rho([\mathcal{P}])$. As a consequence, there may be several "predecessors" of an equivalence class $[\mathcal{P}]$. However, we can define $\mathsf{tail}_\sigma([\mathcal{P}]) = \mathsf{tail}_\sigma(\mathcal{P})$, $\mathsf{tail}_\delta([\mathcal{P}]) = \mathsf{tail}_\delta(\mathcal{P})$ and $\mathcal{L}([\mathcal{P}]) = \mathcal{L}(\mathcal{P})$. In the sequel, we use $\mathcal{P}$ instead of $[\mathcal{P}]$ whenever it is clear from the context.

In a tree-shaped structure where each node has a unique predecessor, each path $\mathcal{P}$ is identical to its equivalence class $[\mathcal{P}]$. This no longer holds for the general graph structure. The notion of paths in a frame is needed to define cycles which are crucial to establish termination condition when building a frame.

Definition 8 (cycle). *Let $\mathcal{F} = \langle \mathcal{N}, \mathcal{N}_o, \Omega, \delta \rangle$ be a frame for a $\mathcal{SHOIQ}_{(+)}$ knowledge base $(\mathcal{T}, \mathcal{R})$ with a set $\mathscr{P}$ of paths in $\mathcal{F}$. Let $\mathscr{R}$ be a set of pairs $(\mathcal{P}_r, \xi_r)$, called* root paths*, where $\mathcal{P}_r \in \mathscr{P}$ and $\xi_r \subseteq \xi(\mathsf{tail}_\sigma(\mathcal{P}_r))$. Let Θ be a set of quadruples $(\mathcal{P}, \rho, \mathcal{Q}, \nu)$ where $\mathcal{P}, \mathcal{Q} \in \mathscr{P}$ ($\mathcal{P} \neq \mathcal{Q}$), respectively called* cycled *and* cycling *paths of Θ , $\rho \in \xi(\mathsf{tail}_\sigma(\mathcal{P}))$, $\nu \in \xi(\mathsf{tail}_\sigma(\mathcal{Q}))$, respectively called* cycled *and* cycling *rays of Θ. A ρ-neighbor of a cycled (*resp. *cycling) path $\mathcal{P}$ is a cycled (*resp. *cycling) neighbor of $\mathcal{P}$ if ρ is a cycled (*resp. *cycling) ray of $\mathcal{P}$. We say that Θ is a* cycle *w.r.t. a set $\mathscr{R}$ of root paths if for each quadruple $(\mathcal{P}, \rho, \mathcal{Q}, \nu) \in \Theta$ the following conditions are satisfied:*

1. *$o \notin \mathcal{L}(\mathcal{P}) \cup \mathcal{L}(\mathcal{Q}) \cup \bigcup_{\rho \in \xi(\mathsf{tail}_\sigma(\mathcal{P})) \cup \xi(\mathsf{tail}_\sigma(\mathcal{Q}))} l(\rho)$ for all $o \in \mathbf{C}_o$;*
2. *$\mathcal{L}(\mathcal{P}) = l(\nu)$, $\mathcal{L}(\mathcal{Q}) = l(\rho)$ and $r(\rho) = r^-(\nu)$.*
3. *for each ray $\rho' \in \xi(\mathsf{tail}_\sigma(\mathcal{P}))$ that is not cycled, there are a sequence $\mathcal{P}_1, \cdots, \mathcal{P}_n \in \mathscr{P}$, some $(\mathcal{P}_0, \rho_0, \mathcal{Q}_0, \nu_0) \in \Theta$ and a root path $(\mathcal{P}_r, \xi_r) \in \mathscr{R}$ such that $\mathcal{P}_i \neq \mathcal{P}_j$ for $1 \leq i < j \leq n$, $\mathcal{P}_1 = \mathcal{P}$, $\mathcal{P}_2$ is the ρ'-neighbor of $\mathcal{P}_1$, $\mathcal{P}_{i+1}$ is a neighbor of $\mathcal{P}_i$ for $1 \leq i < n$, $\mathcal{P}_k = \mathcal{Q}_0$ for some $1 < k < n-1$, and $\mathcal{P}_n = \mathcal{P}_r$, $\mathcal{P}_{n-1}$ is a ρ_r-neighbor of $\mathcal{P}_n$ with $\rho_r \in \xi_r$.*
4. *for each ray $\nu' \in \xi(\mathsf{tail}_\sigma(\mathcal{Q}))$ that is not cycling and each sequence $\mathcal{P}_1, \cdots, \mathcal{P}_n \in \mathscr{P}$ such that $\mathcal{P}_i \neq \mathcal{P}_j$ for $1 \leq i < j \leq n$, $\mathcal{P}_1 = \mathcal{Q}$, $\mathcal{P}_2$ is the ν'-neighbor of $\mathcal{Q}$, and $\mathcal{P}_{i+1}$ is a neighbor of $\mathcal{P}_i$ for $1 \leq i < n$, there is some $(\mathcal{P}_0, \rho_0, \mathcal{Q}_0, \nu_0) \in \Theta$ such that one of the following conditions is satisfied:*
   (a) *there is some $1 < k \leq n$ with $\mathcal{P}_k = \mathcal{Q}_0$ or $\mathcal{P}_k = \mathcal{P}_0$, and $\mathcal{P}_i$ is not a cycling and cycled neighbor for all $1 \leq i \leq k$;*
   (b) *there are $\mathcal{P}_{n+1}, \cdots, \mathcal{P}_{n+m} \in \mathscr{P}$ with $\mathcal{P}_0 = \mathcal{P}_{n+m}$ or $\mathcal{Q}_0 = \mathcal{P}_{n+m}$ such that $\mathcal{P}_i \neq \mathcal{P}_j$ for $1 \leq i < j \leq n+m$, $\mathcal{P}_{i+1}$ is a neighbor of $\mathcal{P}_i$ for all $n \leq i < n+m$, and $\mathcal{P}_i$ is not a cycling and cycled neighbor for all $1 \leq i \leq n+m$;*

We use $\mathscr{R}_0$ to denote the set of all pairs $(\mathcal{P}_r, \xi(\mathsf{tail}_\sigma(\mathcal{P}_r)))$ where $\mathcal{P}_r$ is a nominal path. A primary cycle *Θ_0 is a cycle w.r.t. $\mathscr{R}_0$. Furthermore, we define a* reachable cycle *Θ' of a cycle of Θ if Θ' is a cycle w.r.t. the set of all pairs $(\mathcal{P}_r, \xi_r)$ where $\mathcal{P}_r$ is a cycled path of Θ and ξ_r is the set of all cycled rays of $\mathcal{P}_r$.*

Note that a cycle Θ may encapsulate a *loop* if it includes two quadruples $(\mathcal{P}, \rho, \mathcal{Q}, \nu)$, $(\mathcal{P}', \rho', \mathcal{Q}', \nu')$ such that $\mathcal{Q}'$ is a reachable path of $\mathcal{Q}$ via ρ. A loop can be formed from a sequence $\mathcal{P}_1, \cdots, \mathcal{P}_n \in \mathscr{P}$ $(n > 3)$ such that $\mathcal{P}_1 = \mathcal{P}_n$, $\mathcal{P}_i \neq \mathcal{P}_j$ for $1 \leq i < j < n$ and $\mathcal{P}_{i+1}$ is a neighbor of $\mathcal{P}_i$ for $1 \leq i < n$). Moreover, it is possible that there are two quadruples $(\mathcal{P}, \rho, \mathcal{Q}, \nu), (\mathcal{P}', \rho', \mathcal{Q}', \nu') \in \Theta$ such that $\mathcal{Q}' = \mathcal{Q}$, $\nu = \nu'$ and $\mathcal{P}' \neq \mathcal{P}$, $\rho \neq \rho'$, or $\mathcal{P}' = \mathcal{P}$, $\rho = \rho'$ and $\mathcal{Q}' \neq \mathcal{Q}$, $\nu \neq \nu'$.

Intuitively, a (primary) cycle allows one to "cut" all paths started from nominal paths of a frame into two parts : the first path which is connected to nominal paths is not replicated while the second part can be infinitely lengthened. Condition 1, Definition 8 says that a cycle should not include nominal star-types which must not replicated. Condition 2 says that a cycled path "matches" its cycling path via a ray with the same label. Condition 3 not only provides the relationship between two paths $\mathcal{P}, \mathcal{Q}$ for each $(\mathcal{P}, \rho, \mathcal{Q}, \nu) \in \Theta$ but also ensures that all *non-cycled* neighbors of each $\mathcal{P}$ are filled in a cycle. Condition 4 ensures that an extension of cycled paths $\mathcal{P}$ via their cycled neighbors is possible by replicating paths from its cycling path $\mathcal{Q}$ via cycling rays.

As a consequence, the existence of a cycle allows one to "unravel" a set $\mathscr{P}$ of paths in a frame to obtain a possibly infinite set $\widehat{\mathscr{P}}$ of paths. The following lemma characterizes this crucial property and provides a bound on the size of a cycle.

Lemma 3. *Let $\mathcal{F} = \langle \mathcal{N}, \mathcal{N}_o, \Omega, \delta \rangle$ be a frame for a $\mathcal{SHOIQ}_{(+)}$ knowledge base $(\mathcal{T}, \mathcal{R})$. Let Θ be a cycle in $\mathcal{F}$.*

1. *There exists an extension $\widehat{\mathscr{P}_\Theta}$ of paths between cycled and cycling paths such that each path $\mathcal{P}_0 \in \widehat{\mathscr{P}_\Theta}$ has exactly $|\xi(\mathsf{tail}_\sigma(\mathcal{P}_0))|$ neighbors.*
2. *If Θ' is a reachable cycle of Θ then $|\Theta'| \leq |\Theta| \times |\xi|^{2^\ell}$ where $|\xi|$ is the maximal number of rays of a star-type, and $\ell = 2^{2 \times |\mathbf{CL}(\mathcal{T},\mathcal{R})| \times |\mathbf{R}(\mathcal{T},\mathcal{R})|}$.*

A proof of Lemma 3 can be based on the fact that all paths between cycling and cycled paths of a cycle do not cross the borders defined by the cycle. Therefore, these paths can be replicated and pasted to cycled paths. With regard to the size of a cycle, we can use the following construction: each path starts from a nominal star-type in $\mathcal{N}_o$ and is lengthened through star-types (more precisely, through layers of rays of star-types). We define inductively a level n of a path $\mathcal{P}$ as follows: (i) all nominal paths are at level 0, (ii) a path $\mathcal{P}'$ is at level $i+1$ if it has a neighbor at level i, and all neighbors of $\mathcal{P}'$ are at a level which are equal or greater than i. This implies that there are no two neighbor paths which are located on two levels whose difference is greater than 1.

Assume that there is a pair of paths $(\mathcal{Q}, \mathcal{Q}')$ such that $\mathcal{Q}$ is at level $i > 1$ and $\mathcal{Q}'$ is a ν-neighbor of $\mathcal{Q}$ at level $i-1$ iff there is a pair of paths $(\mathcal{P}, \mathcal{P}')$ such that $\mathcal{P}$ is at level $j > i$, $\mathcal{P}'$ is a ρ-neighbor of $\mathcal{P}$ at level $j+1$, and $\mathcal{L}(\mathcal{Q}) = \mathcal{L}(\mathcal{P}')$, $\mathcal{L}(\mathcal{Q}') = \mathcal{L}(\mathcal{P})$, $r(\nu) = r^-(\rho)$. This implies that all such quadruples $(\mathcal{P}, \rho, \mathcal{Q}, \nu)$ can form a cycle. Moreover, there are at most ℓ different labels of pairs $(\mathcal{Q}, \nu)$. This implies that one cycle can be detected after creating at most 2^ℓ levels. Thus, we have $|\Theta'| \leq |\Theta| \times |\xi|^{2^\ell}$ where $|\xi|$ is the maximal number of rays of star-type. A more complete proof of Lemma 3 can be found in [14].

Let Θ be a cycle in a frame. Definition 8 ensures that each reachable path of some path $\mathcal{Q}$ with $(\mathcal{P}, \rho, \mathcal{Q}, \nu) \in \Theta$ goes through a star-type $\sigma = \mathsf{tail}_\sigma(\mathcal{P}')$ with some

$(\mathcal{P}', \rho', \mathcal{Q}', \nu') \in \Theta$. As mentioned in Lemma 3, such a cycle allows one to "unravel" infinitely the frame to obtain a model of a KB in $\mathcal{SHOIQ}$ (without transitive closure of roles). However, such a cycle structure is not sufficient to represent models of a KB with transitive closure of roles since a concept such as $\exists Q^{\oplus}.D \in \mathcal{L}(\mathcal{P})$ can be satisfied by a Q-reachable path $\mathcal{P}'$ of $\mathcal{P}$ which is arbitrarily far from $\mathcal{P}$. There are the following possibilities for an algorithm which builds a frame: (i) the algorithm stops building the frame as soon as a cycle Θ is detected such that each concept of the form $\exists Q^{\oplus}.D$ occurring in $\mathcal{L}(\mathcal{P})$ is satisfied for each cycled path $\mathcal{P}$ of Θ, i.e., $\mathcal{P}$ has a Q-reachable path $\mathcal{P}'$ with $\exists Q.D \in \mathcal{L}(\mathcal{P})$, (ii) despite of several detected cycles, the algorithm continues building the frame until each concept of the form $\exists Q^{\oplus}.D$ occurring in $\mathcal{L}(\mathcal{P})$ is satisfied for each cycled path $\mathcal{P}$ of Θ. If we adopt the first possibility, the completeness of such an algorithm cannot be established since there are models in which paths satisfying concepts of the form $\exists Q^{\oplus}.D$ can spread over several "iterative structures" such as cycles. For this reason, we adopt the second possibility by introducing into frames an additional structure, namely *blocking-blocked cycles*, which determines a sequence of cycles $\Theta_1, \cdots, \Theta_k$ such that Θ_{i+1} is a reachable cycle of Θ_i for satisfying concepts of the form $\exists Q^{\oplus}.D$.

Definition 9 (blocking). *Let $\mathcal{F} = \langle \mathcal{N}, \mathcal{N}_o, \Omega, \delta \rangle$ be a frame for a $\mathcal{SHOIQ}_{(+)}$ knowledge base $(\mathcal{T}, \mathcal{R})$ with a set $\mathscr{P}$ of paths in $\mathcal{F}$. A cycle Θ' is* blocked *by a cycle Θ if there are cycles $\Theta_1, \cdots, \Theta_k$ with $\Theta = \Theta_1$, $\Theta' = \Theta_k$ such that Θ_{i+1} is a reachable cycle of Θ_i for $1 \leq i < k$, and the following conditions are satisfied:*

1. *For each $1 \leq i < k$, there is no cycle Θ'' such that*
   (a) *Θ'' is a reachable cycle of Θ_i and Θ_{i+1} is a reachable cycle of Θ'', and*
   (b) *For each $(\mathcal{P}, \rho, \mathcal{Q}, \nu) \in \Theta''$ and each concept $\exists Q^{\oplus}.D \in \mathcal{L}(\mathcal{P})$, $\mathcal{P}$ has a Q-reachable path $\mathcal{P}'$ via a non cycled ray with $\exists Q.D \in \mathcal{L}(\mathcal{P}')$ iff the ν-neighbor $\mathcal{Q}'$ of $\mathcal{Q}$ has a Q-reachable path $\mathcal{Q}''$ via a non cycling ray with $\exists Q.D \in \mathcal{L}(\mathcal{Q}'')$.*
2. *For each $(\mathcal{P}_k, \rho_k, \mathcal{Q}_k, \nu_k) \in \Theta_k$, there is some $(\mathcal{P}_1, \rho_1, \mathcal{Q}_1, \nu_1) \in \Theta_1$ such that*
   (a) *$\mathcal{L}(\mathcal{P}_1) = \mathcal{L}(\mathcal{P}_k)$, $\mathcal{L}(\mathcal{Q}_1) = \mathcal{L}(\mathcal{Q}_k)$, $r(\rho_1) = r(\rho_k)$, and*
   (b) *If there is a concept $\exists Q^{\oplus}.D \in \mathcal{L}(\mathcal{P}_k)$ such that the path $\mathcal{P}_k$ has no Q-reachable path $\mathcal{P}'$ with $\exists Q.D \in \mathcal{L}(\mathcal{Q}')$ then the path $\mathcal{Q}_1$ has a Q-reachable path $\mathcal{Q}$ such that the two following conditions are satisfied:*
      i. *$\exists Q.D \in \mathcal{L}(\mathcal{Q})$, or $\mathcal{Q}$ has a Q-reachable path $\mathcal{Q}'$ with $\exists Q.D \in \mathcal{L}(\mathcal{Q}')$,*
      ii. *there are $(\mathcal{P}_j, \rho_j, \mathcal{Q}_j, \nu_j) \in \Theta_j$, $(\mathcal{P}_{j+1}, \rho_{j+1}, \mathcal{Q}_{j+1}, \nu_{j+1}) \in \Theta_{j+1}$ with some $1 \leq j < k$ such that $\mathcal{Q}'$ is a reachable path of $\mathcal{Q}_j$ and $\mathcal{Q}_{j+1}$ is a reachable path of $\mathcal{Q}'$.*

   In this case, we say that the path $\mathcal{P}_k$ is blocked by the path $\mathcal{Q}_1$ via the ray ρ_k. $\lhd$

Definition 9 provides an exact structure of a frame in which blocked paths can be detected. Such a frame contains sequentially reachable cycles between a blocking cycle Θ_1 and its blocked cycle Θ_k, which allows for unravelling the frame between Θ_k and Θ_1, and satisfying all concepts of the form $\exists Q^{\oplus}.D$ in the labels of paths in Θ_1. Condition 1 ensures that there is no useless cycle for the satisfaction of concepts $\exists Q^{\oplus}.D$ which is located between two cycles Θ_i and Θ_i with $i < k$. For a concept $\exists Q^{\oplus}.D \in \mathcal{L}(\mathcal{P}_k)$ that is not satisfied from the path $\mathcal{P}_k$ to all existing paths (i.e. it is

not satisfied in the "past"), it must be satisfied from $\mathcal{P}_k$ to paths that are devised by unravelling (i.e. it is satisfied in "the future"). Therefore, it is required that such concepts $\exists Q^{\oplus}.D$ are satisfied in the "future" from the blocking path $\mathcal{P}_1$ of $\mathcal{P}_k$ (Condition 2, Definition 9). Moreover, for a concept $\exists Q^{\oplus}.D \in \mathcal{L}(\mathcal{P})$ that is not satisfied in the "past", either it is satisfied from $\mathcal{P}$ to some paths that are explicitly added to the frame, or it is propagated to a some blocked path thanks to Property 11, Definition 4.

Remark 2. The constant k mentioned in Definition 9 depends to the number of distinct ray labels (i.e. the triple $\langle \mathcal{L}(\mathcal{P}), r(\rho), l(\rho)\rangle$ for each ray $\rho \in \mathsf{tail}_\sigma(\mathcal{P}))$ occurring a blocking cycle Θ_1 and the number of concepts $\exists Q^{\oplus}.D$ occurring in each cycling path label in Θ_1. Since the number of distinct ray labels is bounded by ℓ (Lemma 3) and the number of concepts $\exists Q^{\oplus}.D$ occurring in each cycling path label is bounded by $\mathbf{CL}(\mathcal{T}, \mathcal{R})$, we have k is bounded by $2 \times \ell$ where $\ell = 2^{2\times|\mathbf{CL}(\mathcal{T},\mathcal{R})|\times|\mathbf{R}(\mathcal{T},\mathcal{R})|}$.

Definition 10 (valid frame). *Let $(\mathcal{T}, \mathcal{R})$ be a $\mathcal{SHOIQ}$ knowledge base. A frame $\mathcal{F} = \langle \mathcal{N}, \mathcal{N}_o, \Omega, \delta\rangle$ with a set $\mathscr{P}$ of paths is* valid *if the following conditions are satisfied:*

1. *For each nominal $o \in \mathbf{C}_o$, there is a unique $\sigma_o \in \mathcal{N}_o$ such that $o \in \lambda(\sigma_o)$ and $\delta(\sigma_o) = 1$;*
2. *For each star-type $\sigma \in \mathcal{N}$, σ is valid.*
3. *If $\exists Q^{\oplus}.C \in \mathcal{L}(\mathcal{P}_0)$ for some $\mathcal{P}_0 \in \mathscr{P}$ then there are $\mathcal{P}, \mathcal{P}' \in \mathscr{P}$ such that one of the following conditions is satisfied:*
   (a) $\mathcal{P}_0 = \mathcal{P} = \mathcal{P}'$ and $\exists Q.C \in \mathcal{L}(\mathcal{P}_0)$;
   (b) $\mathcal{P}'$ is a Q-reachable of $\mathcal{P}$, and $\exists Q.C \in \mathcal{L}(\mathcal{P}')$ where $\mathcal{P} = \mathcal{P}_0$ or $\mathcal{P}$ blocks $\mathcal{P}_0$;
   (c) $\mathcal{P}$ is a $Q^{\ominus}$-reachable of $\mathcal{P}'$, and $\exists Q.C \in \mathcal{L}(\mathcal{P}')$ where $\mathcal{P} = \mathcal{P}_0$ or $\mathcal{P}$ blocks $\mathcal{P}_0$. ◁

Conditions 1-3 in Definition 10 ensure the satisfaction of tableau properties in Definition 3. Note that Condition 1 is compatible with the fact that cycles in a frame never consist of nominal star-types (Definition 8). In particular, Condition 3 provides the satisfaction of concepts $\exists Q^{\oplus}.D$ occurring in the labels of paths thanks to the blocking condition introduced in Definition 9.

We now present Algorithm 1 for building a valid frame. This algorithm starts by adding nominal star-types to the frame. For each non blocked path $\mathcal{P}$ with a ray $\rho \in \xi(\mathsf{tail}_\sigma(\mathcal{P}))$ such that $\delta(\mathsf{tail}_\sigma(\mathcal{P}), \rho) = \delta(\mathsf{tail}_\sigma(\mathcal{P})) + 1$, the algorithm picks in a nondeterministic way a valid star-type ω that matches $\mathsf{tail}_\sigma(\mathcal{P})$ via ρ, and updates the values $\Omega(\mathsf{tail}_\sigma(\mathcal{P}), \rho)$, $\Omega(\omega, \rho')$, $\delta(\mathsf{tail}_\sigma(\mathcal{P}), \rho)$, $\delta(\omega, \rho')$, eventually, $\delta(\mathsf{tail}_\sigma(\mathcal{P}))$ and $\delta(\omega)$ by calling $\mathsf{updateFrame}(\cdots)$. The algorithm terminates when a blocked cycle is detected. To check the blocking condition, the algorithm can compare $\mathscr{R}_i$ for each new level i of rays with each $\mathscr{R}_j$ for all $j < i$ (the notion of levels of rays in a frame is given in the proof of Lemma 3) where $\mathscr{R}_j$ is denoted for the set of different ray labels at level i. If $\mathscr{R}_j = \mathscr{R}_i$ and the last cycle that was detected located at some level $l < j$, then a new (reachable) cycle from level j to i is formed.

Figure 2 depicts a frame when executing Algorithm 1 for $\mathcal{K}_1$ in the example presented in Section 1. The algorithm builds a frame $\mathcal{F} = \langle \mathcal{N}, \mathcal{N}_o, \Omega, \delta\rangle$ where $\mathcal{N} = \{\sigma_0, \sigma_1, \sigma_2, \sigma_3, \sigma_4\}$ and $\mathcal{N}_o = \{\sigma_0\}$. The dashed arrows indicate how the function $\Omega(\sigma, \rho)$ can be built. For example, $\Omega(\sigma_0, \rho_0) = \{(\sigma_1, \nu_0, 1)\}$, $\Omega(\sigma_0, \rho_1) = \{(\sigma_2, \rho'_0, 1)\}$

Require: A $\mathcal{SHOIQ}_{(+)}$ knowledge base $(\mathcal{T}, \mathcal{R})$
Ensure: A frame $\langle \mathcal{N}, \mathcal{N}_o, \Omega, \delta \rangle$ for $(\mathcal{T}, \mathcal{R})$
1: Let Σ be the set of all star-types for $(\mathcal{T}, \mathcal{R})$
2: **for all** $o \in \mathbf{C}_o$ **do**
3: **if** there is no $\sigma \in \mathcal{N}$ such that $o \in \lambda(\sigma)$ **then**
4: Choose a star-type $\sigma_o \in \Sigma$ such that $o \in \lambda(\sigma_o)$
5: Set $\delta(\sigma_o) = 1$, $\mathcal{N} = \mathcal{N} \cup \{\sigma_o\}$ and $\mathcal{N}_o = \mathcal{N}_o \cup \{\sigma_o\}$
6: Set $\delta(\sigma_o, \rho) = 0$, $\Omega(\sigma_o, \rho) = \emptyset$ for all $\rho \in \xi(\sigma_o)$
7: **end if**
8: **end for**
9: **while** there is a path $\mathcal{P}$ that is not blocked and a ray $\rho \in \xi(\mathsf{tail}_\sigma(\mathcal{P}))$ such that $\mathsf{tail}_\delta(\mathcal{P}) = \delta(\mathsf{tail}_\sigma(\mathcal{P}), \rho) + 1$ **do**
10: Choose a star-type $\sigma' \in \Sigma$ such that there is a ray $\rho' \in \xi(\sigma')$ satisfying $l(\rho) = \lambda(\sigma')$, $l(\rho') = \lambda(\sigma)$, $r(\rho') = r^-(\rho)$, and $\sigma' \in \mathcal{N}$ implies $\delta(\sigma') = \delta(\sigma', \rho') + 1$ or $\delta(\sigma') = \delta(\sigma', \rho'')$ for all $\rho'' \in \xi(\sigma')$
11: $\mathsf{updateFrame}(\sigma, \rho, \sigma', \rho')$
12: **end while**

Algorithm 1. An algorithm for building a frame

Require: A star-type $\sigma \in \mathcal{N}$ in a frame $\mathcal{F} = \langle \mathcal{N}, \mathcal{N}_o, \Omega, \delta \rangle$ with a ray $\rho \in \xi(\sigma)$, and a new star-type σ' with a ray $\rho' \in \xi(\sigma')$ such that $l(\rho) = \lambda(\sigma')$, $l(\rho') = \lambda(\sigma)$, $r(\rho') = r^-(\rho)$
Ensure: $\mathsf{updateFrame}(\sigma, \rho, \sigma', \rho')$
1: **if** there exists a star-type $\omega \in \mathcal{N}$ such that ω is equivalent to σ' **then**
2: Set $\delta(\sigma, \rho) = \delta(\sigma, \rho) + 1$
3: Let $\nu \in \xi(\omega)$ such that $r(\nu) = r(\rho')$ and $l(\nu) = l(\rho')$
4: **if** $\delta(\omega, \nu) == \delta(\omega)$ **then**
5: Set $\delta(\omega) = \delta(\omega) + 1$
6: **end if**
7: Set $\delta(\omega, \nu) = \delta(\omega, \nu) + 1$
8: $\mathsf{add}(\Omega(\omega, \nu), (\sigma, \rho, \delta(\sigma, \rho)))$
9: $\mathsf{add}(\Omega(\sigma, \rho), (\omega, \nu, \delta(\omega, \nu)))$
10: **else**
11: Add σ' to $\mathcal{N}$
12: Set $\delta(\sigma, \rho)$=$\delta(\sigma, \rho) + 1$
13: Set $\delta(\sigma') = 1$, $\delta(\sigma', \rho') = 1$ and $\Omega(\sigma', \rho') = \{(\sigma, \rho, \delta(\sigma, \rho))\}$
14: Set $\delta(\sigma', \rho'') = 0$ and $\Omega(\sigma', \rho'') = \emptyset$ for all $\rho'' \neq \rho'$
15: $\mathsf{add}(\Omega(\sigma, \rho), (\sigma', \rho', 1))$
16: **end if**

Algorithm 2. $\mathsf{updateFrame}(\sigma, \rho, \sigma', \rho')$ updates $\mathcal{F}$ when adding σ' to $\mathcal{N}$

where ρ_0 and ρ_1 are the respective horizontal and vertical rays of σ_0; ν_0 is the left ray of σ_1; ρ'_0 is the vertical ray of σ_2. Moreover, the directed dashed arrow from σ_0 to σ_1 indicates that the ray ρ_0 of σ_0 can match the ray ν_0 on the left ray of σ_1 since $l(\rho_0) = \lambda(\sigma_1)$, $r(\nu_0) = \lambda(\sigma_0)$, $r(\nu_0) = r^-(\rho_0)$.

The algorithm generates $\delta(\sigma_0) = 1$, $\delta(\sigma_1) = 1$, $\delta(\sigma_2) = 1$ and forms a cycle Θ consisting of the following quadruples : $((\sigma_3, 3), \rho_1, (\sigma_3, 2), \rho_2)$ (ρ_1 and ρ_2 are the right and left rays of σ_3, respectively) and $((\sigma_4, 2), \rho_3, (\sigma_4, 1), \rho_4)$ (ρ_3 and ρ_4 are the right and left rays of σ_4 respectively). Note that for the sake of brevity, we use just $\mathsf{tail}_\sigma(\mathcal{P})$ and $\mathsf{tail}_\delta(\mathcal{P})$ to denote a path in the quadruples.

The cycle Θ is blocked since all concepts $\exists S^+.\{o\}$ occurring in cycled paths are satisfied. A model of the ontology can be built by starting from σ_0 and getting (i) σ_4 via σ_1, (ii) σ_3 via σ_1, and (iii) σ_3 via σ_2. From σ_3 and σ_4, the model goes through σ_3 and σ_4 infinitely. Note that from any individual x satisfying σ_3 (or σ_4), i.e. the "label" of x contains $\exists Q^+.\{o\}$, there is a path containing S which goes back the individual satisfying σ_0. Thus, the concept $\exists Q^+.\{o\}$ is satisfied for each individual whose label contains $\exists Q^+.\{o\}$.

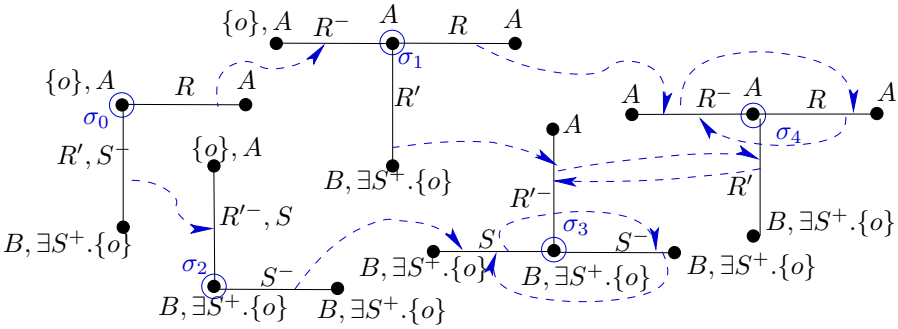


Fig. 2. A frame obtained by Algorithm 1 for $\mathcal{K}_1$ in the example in Section 1

Lemma 4. *Let $(\mathcal{T}, \mathcal{R})$ be a $\mathcal{SHOIQ}_{(+)}$ knowledge base.*

1. *Algorithm 1 terminates.*
2. *If Algorithm 1 can build a valid frame for $(\mathcal{T}, \mathcal{R})$ then there is a tableau for $(\mathcal{T}, \mathcal{R})$.*
3. *If there is a tableau for $(\mathcal{T}, \mathcal{R})$ then Algorithm 1 can build a valid frame $\mathcal{F}$ for $(\mathcal{T}, \mathcal{R})$.*

Proof (sketch). Let Θ_k be a blocked cycle by Θ_1. According to Remark 2, k is bounded by $\mathcal{O}(2^{|(\mathcal{T},\mathcal{R})|})$. Moreover, after eliminating "useless cycles" between two cycles Θ_i and Θ_{i+1} for $1 \leq i < k$ according to Condition 1, Definition 9 the number of useful cycles between Θ_i and Θ_{i+1} is bounded by $\mathcal{O}(2^{2^{|(\mathcal{T},\mathcal{R})|}})$. This implies that Algorithm 1 can add at most a triple exponential number of paths to the frame to form a blocked cycle. For the soundness of Algorithm 1, we can extend the set $\mathscr{P}$ of paths to a set $\widehat{\mathscr{P}}$ of extended paths by "unravelling" the frame between blocking-blocked cycles. The set $\widehat{\mathscr{P}}$ allows one to satisfy concepts $\exists Q^{\oplus}.D$ in blocked paths which are not satisfied in the

"past". Moreover, a concept $\exists Q^{\oplus}.D$ of a path that is not satisfied in the "past" will be propagated to a blocked path via a Q-path. Therefore, it will be satisfied in $\widehat{\mathscr{P}}$. Unlike the unravelling of a completion graph for $\mathcal{SHOIQ}$ where there is no loop in the model, the unravelling of a frame may yield an infinite number of loops in the model. Note that the unravelling of a frame replicates cycles which may encapsulate loops.

Regarding completeness, we first reduce a tableau to a frame that does not contain any useless cycle. Then, we use the obtained frame to guide the algorithm (i) to choose valid star-types, (ii) to ensure that $\delta(\sigma) = 1$ for each nominal star-type σ, and (iii) to detect a pair (Θ_1, Θ_k) of blocking and blocked cycles as soon as some "representative" concepts of the form $\exists Q^{\oplus}.D$ in Θ_1 are satisfied. We refer the readers to [14] for a complete proof of Lemma 4. □

The following theorem is a consequence of Lemma 4.

Theorem 1. *The problem of consistency for $\mathcal{SHOIQ}_{(+)}$ can be decided in non-deterministic triply exponential time in the size of a $\mathcal{SHOIQ}_{(+)}$ knowledge base.*

4 Optimizing The Algorithm

The algorithm for deciding the consistency of a $\mathcal{SHOIQ}_{(+)}$ knowledge base (Algorithm 1) uses at most a doubly exponential number of star-types to build a frame. This is due to the fact that numbers are encoded in binary, that is, a star-type may have an exponential number of rays. Pratt-Hartmann [9] has shown that it is possible to use an exponential number of star-types to represent a model of a KB in $\mathcal{C}^2$ which is slightly different from $\mathcal{SHOIQ}$ in terms of expressiveness. If we can transfer this method to $\mathcal{SHOIQ}$ for compressing star-types, it would be applied to $\mathcal{SHOIQ}_{(+)}$ since the number of star-types in a frame does not depend on the presence of transitive closure of roles.

Another technique presented in [15] can be used to reduce non-determinisms due to the choice of valid star-types. Instead of guessing a valid star-type from a set of valid star-types, this technique allows one to build a star-type σ by applying expansion rules to concepts in the core label of σ. Hence, when a star-type σ is transformed into σ' by an expansion rule, an algorithm that implements this technique has to update not only $\Omega(\sigma', \rho')$ and $\delta(\sigma')$ but also $\Omega(\sigma'', \rho'')$ and $\delta(\sigma'')$ for each neighbor σ'' of σ and σ' (σ'' is a neighbor of σ' if there is some $(\sigma'', \rho'', d'') \in \Omega(\sigma', \rho')$). These updates must ensure that each path which has got through σ can now get through σ'. This process of changes can spread over neighbors of σ'' and so on.

With regard to blocking, the technique presented in [15] can take advantage of a specific structure of frames for $\mathcal{SHOIQ}$ to design an efficient algorithm for checking blocking condition. This structure consists of partitioning star-types into layers. Although such a structure of frames cannot be maintained for $\mathcal{SHOIQ}_{(+)}$, paths in a frame for $\mathcal{SHOIQ}_{(+)}$ would allow us to achieve the same behavior.

5 Conclusion

In this paper, we have presented a decision procedure for the description logic $\mathcal{SHOIQ}$ with transitive closure of roles in concept axioms, whose decidability was not known.

The most significant feature of our contribution is to introduce a structure based on a new blocking condition for characterizing models which have an infinite non-tree-shaped part. This structure would provide an insight into regularity of such models which would be enjoyed by a more expressive DL, such as $\mathcal{ZOIQ}$ [6], whose decidability remains open. In future work, we aim to improve the algorithm by making it more goal-directed and aim to investigate another open question about the hardness of $\mathcal{SHOIQ}_{(+)}$.

References

1. Patel-Schneider, P., Hayes, P., Horrocks, I.: Owl web ontology language semantics and abstract syntax. In: W3C Recommendation (2004)
2. Tobies, S.: The complexity of reasoning with cardinality restrictions and nominals in expressive description logics. Journal of Artificial Intelligence Research 12, 199–217 (2000)
3. Aho, A.V., Ullman, J.D.: Universality of data retrieval languages. In: Proceedings of the 6th of ACM Symposium on Principles of Programming Language (1979)
4. Baader, F.: Augmenting concept languages by transitive closure of roles: An alternative to terminological cycles. In: Proceedings of the Twelfth International Joint Conference on Artificial Intelligence (1991)
5. Ortiz, M.: An automata-based algorithm for description logics around $\mathcal{SRIQ}$. In: Proceedings of the fourth Latin American Workshop on Non-Monotonic Reasoning 2008, CEUR-WS.org (2008)
6. Calvanese, D., Eiter, T., Ortiz, M.: Regular path queries in expressive description logics with nominals. In: IJCAI, pp. 714–720 (2009)
7. Horrocks, I., Sattler, U.: A tableau decision procedure for $\mathcal{SHOIQ}$. Journal of Automated Reasoning 39(3), 249–276 (2007)
8. Motik, B., Shearer, R., Horrocks, I.: Hypertableau reasoning for description logics. J. of Artificial Intelligence Research 36, 165–228 (2009)
9. Pratt-Hartmann, I.: Complexity of the two-variable fragment with counting quantifiers. Journal of Logic, Language and Information 14(3), 369–395 (2005)
10. Le Duc, C., Lamolle, M.: Decidability of description logics with transitive closure of roles. In: Proceedings of the 23rd International Workshop on Description Logics (DL 2010). CEUR-WS.org (2010)
11. Horrocks, I., Sattler, U., Tobies, S.: Practical reasoning for expressive description logics. In: Ganzinger, H., McAllester, D., Voronkov, A. (eds.) LPAR 1999. LNCS, vol. 1705, pp. 161–180. Springer, Heidelberg (1999)
12. Baader, F., Nutt, W.: Basic description logics. In: The Description Logic Handbook: Theory, Implementation and Applications, 2nd edn., pp. 47–104. Cambridge University Press (2007)
13. Fischer, M.J., Ladner, R.I.: Propositional dynamic logic of regular programs. Journal of Computer and System Sciences 18(18), 174–211 (1979)
14. Le Duc, C., Lamolle, M., Curé, O.: A decision procedure for $\mathcal{SHOIQ}$ with transitive closure of roles in concept axioms. In: Technical Report (2013), `http://www.iut.univ-paris8.fr/~leduc/papers/TR-SHOIQTr.pdf`
15. Le Duc, C., Lamolle, M., Curé, O.: An EXPSPACE tableau-based algorithm for $\mathcal{SHOIQ}$. In: Description Logics (2012)

Elastic and Scalable Processing of Linked Stream Data in the Cloud*

Danh Le-Phuoc, Hoan Nguyen Mau Quoc, Chan Le Van, and Manfred Hauswirth

Digital Enterprise Research Institute, National University of Ireland, Galway

Abstract. Linked Stream Data extends the Linked Data paradigm to dynamic data sources. It enables the integration and joint processing of heterogeneous stream data with quasi-static data from the Linked Data Cloud in near-real-time. Several Linked Stream Data processing engines exist but their scalability still needs to be in improved in terms of (static and dynamic) data sizes, number of concurrent queries, stream update frequencies, etc. So far, none of them supports parallel processing in the Cloud, i.e., elastic load profiles in a hosted environment. To remedy these limitations, this paper presents an approach for elastically parallelizing the continuous execution of queries over Linked Stream Data. For this, we have developed novel, highly efficient, and scalable parallel algorithms for continuous query operators. Our approach and algorithms are implemented in our CQELS Cloud system and we present extensive evaluations of their superior performance on Amazon EC2 demonstrating their high scalability and excellent elasticity in a real deployment.

Keywords: Cloud, Linked Data, linked stream processing, continuous queries.

1 Introduction

Realistically, all data sources on the Web are dynamic (across a spectrum). Many current sources are of a slow update nature which is well supported by the existing batch-update infrastructure of Linked Data, e.g., geo-data or DBpedia. However, a fast increasing number of sources produce streams of information for which the processing has to be performed as soon as new data items become available. Examples of such data sources include sensors, embedded systems, mobile devices, Twitter, and social networks, with a steep, exponential growth predicted in the number of sources and the amount of data [22]. Integrating these information streams with other sources will enable a vast range of new "near-real-time" applications. However, due to the heterogeneous nature of the streams and static sources, integrating and processing this data is a difficult and

* This research has been supported by the European Commission under Grant No. FP7-287305 (OpenIoT) and Grant No. FP7-287661 (GAMBAS) and by Science Foundation Ireland under Grant No. SFI/08/CE/I1380 (Lion-II) and Grant No. SFI/12/RC/2289 (INSIGHT).

H. Alani et al. (Eds.): ISWC 2013, Part I, LNCS 8218, pp. 280–297, 2013.

labor-intensive task which gave rise to Linked Stream processing, i.e., extending the notion of Linked Data to dynamic data sources. Linked Stream Data processing engines, such as C-SPARQL [3], EP-SPARQL [1], SPARQL$_{stream}$ [5], and CQELS [19], have emerged as an effort to facilitate this seamless integration of heterogeneous stream data with the Linked Data Cloud using the same abstractions.

None of the above systems could yet systematically and satisfactorily address all scalability aspects existing in Linked Stream Data processing [20], such as the wide range of stream data production frequencies, the size of static data, the number of concurrent queries, elastic load profile support, etc. The existing approaches start to fail when some of these scalability aspects go beyond certain thresholds. For instance, C-SPARQL and EP-SPARQL only can deal with small RDF datasets ($\sim$1 million triples) and most of the approaches are only able consume very slow input stream rates of around 100 triples/second when the number of concurrent queries grows up to 100–1000 [20]. These thresholds are rather modest for real-time and Web applications and must be improved to make the systems usable for practical applications in practical settings, e.g., each car in a city producing a data stream.

On top of this, further scalability issues come from practical limits of computer hardware such as memory size, network bandwidth, processor speed, etc. Even though these parameters will increase over time (Moore's Law), the general argument is likely to remain true for the foreseeable future [17]. However, today's typical computer hardware is cheap and almost indefinitely replicable [17]. The total-cost-of-ownership of 8 off-the-shelf commodity servers with 8 processing cores and 128GB of RAM each is much lower than that of a single system with 64 processors and 1TB of RAM. Therefore, distributing the processing load of a Linked Stream Data processing engine over networked computers is a promising strategy to achieve scalability.

Additionally, the trend to Cloud infrastructures, i.e., renting servers on a "pay-per-use" basis, provides a further argument in favor of this strategy. Amazon EC2, Google Cloud, and Microsoft Azure are prominent examples for this development. Building a Linked Stream Data processing engine running on such an elastic cluster potentially enables the engine to adapt to changing processing loads by dynamically adjusting the number of processing nodes in the cluster at runtime. This "elasticity" is vital for processing stream data due to its fluctuating stream rates (e.g., bursty stream rates) and the unpredictable number of parallel queries (e.g., queries can be registered/unregistered at run-time) which result in hard-to-predict computing loads and resource requirements. To enable elasticity in a Cloud environment with on-demand load profiles, the used algorithms and data access must lend themselves to parallelization or must be re-engineered to achieve this property.

To address the above problems, this paper introduces an elastic execution model based on a comprehensive suite of novel parallelizing algorithms for incremental computing of continuous query operators on Linked Stream Data. We present the CQELS Cloud implementation of this model and provide a

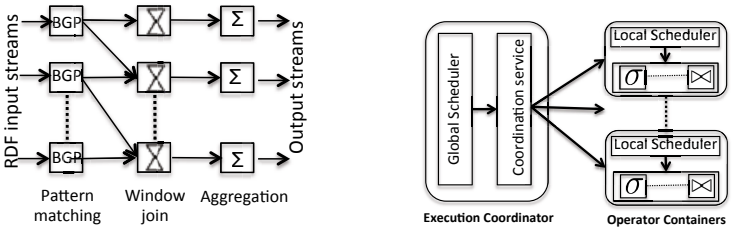
comprehensive evaluation of its scalability and elasticity based on an extensive set of experiments on Amazon EC2. The results show that CQELS Cloud can scale up to throughputs of 100,000s of triples per second for 10,000s of concurrent queries on a cluster of 32 medium EC2 nodes. As we will demonstrate in the paper this is not the ultimate limit of scalability but our approach can scale gracefully to nearly arbitrary loads if more nodes are added.

The remainder of this paper is organized as follows: Section 2 describes our abstract execution model for processing continuous queries over Linked Stream Data on a cluster of networked computers. The required parallelizing algorithms for this execution framework are described in Section 3. Section 4 presents the implementation details of our CQELS Cloud engine and the results of our comprehensive experimental evaluation. We discuss related work in Section 5 and then present our conclusions in Section 6.

2 Elastic Execution of Continuous Queries

This section describes the elastic execution model for processing continuous queries over Linked Stream Data. The execution model is based on the Linked Stream Data model and the query semantics of the CQELS query language (CQELS-QL) [19]. The Linked Stream Data model [3,5,19] is used to model both stream data represented in RDF and static RDF datasets. CQELS-QL is a declarative continuous query language and is a minimal extension of SPARQL 1.1 with additional syntactical constructs to define sliding window operators on RDF data streams.

Our execution model accepts a set of CQELS-QL queries over a set of RDF input streams which produce a set of output streams (RDF streams or relational streams in SPARQL-result formats). These queries will be compiled to a logical query network. The network defines which query algebras the input stream data should go through to return results in the output streams. Figure 1a shows and illustrating example: First triples are extracted from RDF streams by a set of pattern matching operators (basic graph patterns), which then are sent to a number of sliding window joins (derived from the original queries). The triples resulting from these operations are then sent to a set of aggregation operators. The outputs of the aggregation operators are the result streams of the original queries comprising the network.



(a) A continuous query network (b) Elastic execution architecture

Fig. 1. Elastic execution model for Linked Stream Data

Our execution model is based on a distributed architecture as shown in Figure 1b. The logical query network is mapped to a processing network distributed among processing nodes, called Operator Containers (OCs) (see Section 4). The Global Scheduler of the Execution Coordinator uses the Coordination Service to distribute the *continuous processing tasks* to OCs to trigger the corresponding executions concurrently. Similar to Eddies [2,19], the continuous processing tasks are input stream elements associated with *operator signatures* that indicate which physical operators the stream elements need to be processed to satisfy their processing pipeline (mandated by the original queries). Each OC hosts a set of physical query operators that process input streams and forward the output to the consuming operators in the network. The Local Scheduler of an OC is responsible for scheduling the execution of processing tasks assigned by the Global Scheduler to make the best use of computing resources allocated for that OC. This execution architecture supports elasticity by allowing the machines running OCs to join and leave the network dynamically at runtime. The Coordination Service monitors the OC instances for failure or disconnection. In the event that an OC instance leaves, the Coordination Service will notify the Global Scheduler to re-balance / re-assign the "missing" processing to the rest of the network. The Coordination Service maintains all processing state of the whole network whereas each OC only has a mirrored copy of the processing state necessary for its processing tasks. When an OC instance leaves, the Coordination Service will recover its processing state (progress) from the last successful processing state and reassign the tasks to other nodes in the network. When a new OC instance joins, it will notify the Coordination Service of its availability to receive tasks. To start processing assigned tasks, each OC has to synchronize its processing state with the processing state of the Coordination Service. To avoid a single point of failure problem through a failure of the Coordination Service, its processing state is replicated among a set of machines.

Distributing computing over multiple computers supports scalability but also incurs performance costs because bandwidth and latency of the network are several orders of magnitude worse than those of RAM (scalability outweighs the costs as demonstrated by Grids as the predecessor to Clouds). Therefore, to avoid the huge communication overheads of processing raw, "very wordy" RDF streams, we use the dictionary encoding approach of CQELS [19] for compression, i.e., the processing state – input triples, mappings, etc. – is represented as integers. Another way of reducing the communication cost is grouping the operators processing the same inputs into one machine. For example, instead of sending an input triple to multiple machines to apply different triple matching patterns, these triple matching patterns can be grouped to a single machine to avoid sending multiple copies of an input triple (see Section 3.1). Furthermore, the data exchanged among machines is combined into larger units to reduce the additional overhead of packaging and transferring single data items (this is standard "good networking" practise applied in any networking protocol, see Section 4). On the other hand, network latency is much lower than disk latency. Therefore, the performance cost of storing and retrieving data on/from other

nodes in a network is comparable to the cost of local disk access. Thus, data that cannot be stored entirely in memory is distributed to multiple partitions on the disks of multiple computers "in parallel." For instance, the intermediate results from a sub-query to a static data set might have millions of mappings [19] which can be split and indexed (for future search and retrieval operations) on multiple nodes. As a result, the access bandwidth to persistent data can be increased if the number of processing node increases. Interestingly, on typical server hardware, the sequential access to a disk is comparably faster than completely random access to RAM [17]. Storing data to be fetched in sequential blocks and providing distributed indexes to such blocks will overcome the I/O bottleneck of accessing a single disk on a single computer. The availability of data to be searched and fetched can also be increased by increasing the data replicating ratio.

While these strategies may seem overly complicated and heavy, they ensure optimal resource usage, fault tolerance, elasticity, and scalability in line with the accepted state of the art in distributed computing and cloud systems and thus are necessary to achieve our performance and scalability goals.

3 Parallelizing Algorithms for Incremental Evaluation of Continuous Query Operators

The continuous query operators of CQELS-QL, e.g., join, filter, and aggregate [19], are defined as operators on bags of mappings. Each of these operators consumes a set of bags of mappings and returns a bag of mappings which then can be used as intermediate mappings to be consumed in another operator of an execution pipeline, called *upper operator*. In each pipeline, the inputs at the input side are bags of mappings stored in the window buffers of the concerned sliding window operators and the operators return a bag of mappings on the output side. The execution pipelines are continually applied to the input streams. The parallelizing approaches for this continuous evaluation are presented and discussed in the following sections.

3.1 Parallelizing the Incremental Evaluation of Sliding Window Operators

To support maximum throughput and efficiency in a Cloud environment, we designed incremental evaluation strategies for sliding window operators [10] which minimize computational efforts by avoiding re-computations and which can distribute the incremental computing tasks to multiple processing nodes. The incremental computing tasks are triggered by two types of events: the arrival or the expiration of a stream data item, e.g., a mapping. We use a *negative tuple* approach [10,13] to signal expiration events, called *negative mappings*. To implement this, a continuous processing task is assigned to an OC as a mapping with extra information to indicate if it is a negative mapping or a new mapping and to provide operator signatures. The operator signatures are information about

which operator instances should process the mapping [2]. These signatures are used by the Local Scheduler to construct a processing pipeline corresponding to that task.

Stateless operators, e.g., select, project, filter, and triple matching, do not have to maintain a processing state for incremental evaluation, i.e., they do not need to access any previous input. Therefore, parallelizing the incremental evaluation of stateless operators is straight-forward, since a new or expired mapping can be processed directly in parallel to produce the corresponding new or expired mappings. The time necessary for each execution of this kind is usually much shorter than the time spent for transferring an input data item. To save communication cost, we group executions that consume the same input in a single machine. Additionally, grouping executions might improve performance for evaluating concurrent queries. For instance, instead of iterating over all triple patterns to check if an input triple is matched with the constants of these triple patterns, the indexes of such constants can be used to efficiently find which triple patterns have constants being matched with values of the input triple. Grouping operators does not limit the possible degree of parallelization because the processing load is split and distributed per stream data item.

For stateful operators such as join and aggregation, the previous input data have to be consulted to compute the updates necessary when a new or a negative mapping is received. To parallelize the incremental evaluation of these operators, the workers involved in the processing have to be coordinated to share a consistent processing state. How this can be done will be discussed in the next sections. Due to space limitations, we discuss only two operators in the the following which benefit the most from parallelizing their executions: multiway join and aggregation. Their parallel, incremental algorithms are presented below.

3.2 Parallel Multiway Join

Based on an extensive analysis of queries and their executions, we have found that multiway joins are a dominating cost factor in typical queries [18,20]. Inspired by the MJoin approach [23], we developed a parallel multiway join that works over more than two input buffers which are used to store data for sliding windows defined in the continuous execution model introduced in Section 2. As the multiway join is symmetric, without loss of generality, the evaluation of a new mapping μ_1 being inserted into the input buffer R^1 is illustrated in Figure 2. When a mapping μ_1 is inserted into the input buffer R^1, it will be used to probe one of the other input buffers $R^2 \cdots R^n$. Let us assume that R^2 is the next input buffer in the probing sequence. For each mapping μ_2^i in R^2 that is compatible with μ_1, an intermediate joined mapping in the form $\mu_1 \circ \mu_2^i$ is generated. Subsequently, $\mu_1 \circ \mu_2^i$ is recursively used to probe the other input buffers to generate the final mappings. When a buffer that does not return any compatible mapping is found, the probing sequence stops. Following the negative tuple approach [10,13], if a negative mapping arrives, it is used to invalidate expired outputs. This invalidation operation is done by the next in the query pipeline, which consumes the multiway join output as its input buffer.

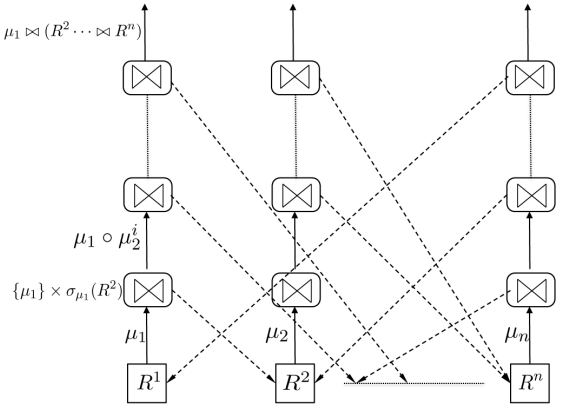


Fig. 2. Multiway join process

The probing sequence and the invalidation operation are triggered by a new mapping or negative mapping and can be executed in parallel on multiple machines given that each machine can access the same set of input buffers. Algorithm 1 shows our incremental evaluation algorithm for the parallel multiway join with n input buffers. Lines 2–5 handle new mappings and line 7 is for forwarding a negative mappings to the upper operator. Lines 2 and 3 synchronize the window $W[i]$ in all machines running this operator. Line 4 checks if the current multiway join is supposed to compute the probing sequence triggered from this new mapping by verifying its operator signatures. Line 5 calls the recursive sub-routine *probingPropagate* (see Algorithm 2) to initialize the probing sequence. The next step in the probing sequence is defined in the sub-routine $findNextProbWin$ in line 2. It chooses the next window buffer $W[i_{next}]$ to be probed to find a compatible mapping to forward to in the next step (line 3). The sub-routine $findNextProbWin$ can be used to inject additional adaptive optimization algorithms for multiway joins [8]. Note that, a multiway join might use buffers to store intermediate mappings from subqueries on static data. They are just a special case of buffers used for sliding windows.

Algorithm 1. Parallel Multi-Way Join

```
   Input: n input buffers W1,..,Wn
1  if a new mapping μ arrives at window W[i] then
2      remove expired tuples from all windows
3      W[i].insert(μ)
4      if μ is assigned for this multiway join instance then
5          probingPropagate(μ,{W[1],..,W[n]}\{W[i]})
6  else
7      propagate negative mapping to upper operator
```

For different window joins which share join predicates over the same RDF streams, we group them into a shared computing network, i.e., a shared join, to reduce network communication overhead as well as to avoid wasting resources due to redundant computation. A shared join has a single execution plan for multiple queries and produces multiple output streams for each separate query involved. The shared join operator consists of two components: the join component and the routing component.

Algorithm 2. Probing propagation *probingPropagate*

```
   Input: μ, k sliding windows {W[i1],..,W[ik]}
1  if k==0 then
2      inext ← findNextProbWin( μ,{W[i1],..,W[ik]})
3      for μ* ∈ W[inext].probe(μ) do
4          probePropagate(μ ∘ μ*, {W1,..,Wk}\{W[inext]})
5  else
6      dispatch μ
```

The join component produces a single intermediate output stream for all queries, and the routing component routes the valid output items to the corresponding output buffer of each continuous query [14]. The join component dominates the query load because the complexity of an m-way join operation is much higher than that of a filtering operation of the routing component for n queries (n is usually smaller than $\prod_{i=1}^{i=m} W_i$ where W_i is the size of buffer i of the m-way join).

To share the computations and the memory when processing multiple joins that have the same set of input buffers, the multiway join algorithm can be used to build a shared join operator, i.e, the multiple join operator. Let us assume m multiple window joins $W_j^1 \cdots \bowtie W_j^m$ where j=1...k and W_j^i is a window buffer extracted from the RDF stream $S^i, i = 1...n$. Let W_{max}^i be the window buffer that has a window size equal to the maximum window size over all $W_j^i, j = 1...k$. Then, the following containment property [14] holds:

$$W_j^1 \cdots \bowtie W_j^n \subseteq W_{max}^1 \cdots \bowtie W_{max}^n$$

Due to this property, the processing of the query $W_{max}^1 \cdots \bowtie W_{max}^n$ produces an output that contains the outputs of all queries $W_j^1 \cdots \bowtie W_j^n, j = 1...k$. Therefore, the join component only has to execute a single multiway query for $W_{max}^1 \cdots \bowtie W_{max}^n$. In the routing component, each resulting mapping then has to be routed to the query that takes it as an input. We call the routing component of the multiple join operator *router*. The router maintains a sorted list of the windows relevant to each join. The windows are ordered by window sizes in increasing order. Each output mapping is checked if its constituent mappings are valid within valid time intervals of the windows of a query. When a mapping satisfies the time condition of the query, it is routed to the query's output buffer.

Figure 3 illustrates a multiple join operator for 3 queries over 2 streams S^1 and S^2 where $Q_1 = W_1^1 \bowtie W_1^2$, $Q_2 = W_2^1 \bowtie W_2^2$ and $Q_3 = W_3^1 \bowtie W_3^2$. This multiple join operator connects the 2-way join operator $W_{max}^1 \bowtie W_{max}^2$ to its router where $W_{max}^1 = W_3^1$ and $W_{max}^2 = W_3^2$. The left-hand side of the figure shows how the router delivers the output mappings from the 2-way join to each query. For instance, when the new mapping $\langle a_1, b_6 \rangle$ arrives at the stream S^1, the 2-way join probes the input buffer W_{max}^2 to generate two output mappings $\langle a_1, b_6, c_1 \rangle$ and $\langle a_1, b_6, c_5 \rangle$. Based on the window conditions of each query, the router routes $\langle a_1, b_6, c_5 \rangle$ to Q_1 and Q_2 and both $\langle a_1, b_6, c_1 \rangle$ and $\langle a_1, b_6, c_5 \rangle$ to Q_3.

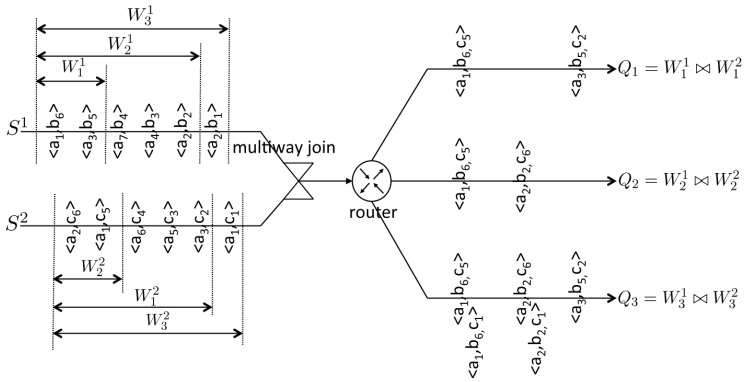


Fig. 3. Shared windows example

Generally, the concurrent queries registered to the system only share subsets of the streams involved in their queries. Therefore, we create a network of multiple join operators to enable sharing of the execution of subqueries for a group of queries. For each group of joins that share the same set of streams a multiple join operator is created. Figure 4 illustrates a network of 4 queries over 4 streams S^1, S^2, S^3 and S^4, where $Q_1 = W_1^1 \bowtie W_1^2 \bowtie W_1^3$, $Q_2 = W_2^2 \bowtie W_2^3$ $Q_3 = W_3^2 \bowtie W_3^3 \bowtie W_3^4$ and $Q_4 = W_4^2 \bowtie W_4^3 \bowtie W_4^4$. This network is composed of three multiple join operators $\bowtie_1^M$, $\bowtie_2^M$ and $\bowtie_3^M$ where $\bowtie_1^M$ is for Q_1, $\bowtie_2^M$ is for Q_2 and $\bowtie_3^M$ for Q_3 and Q_4. Due to space limits, we refer the reader to [18] for a detailed technical discussion of this setting.

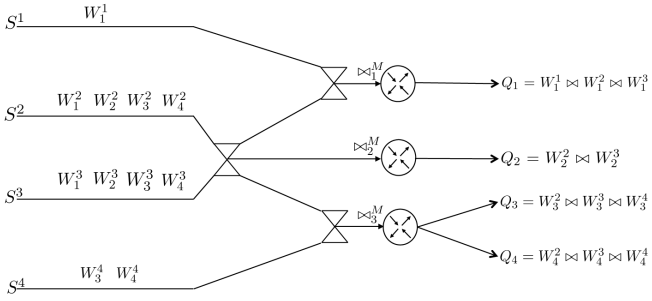


Fig. 4. A network of multiple join operators

3.3 Aggregation

An aggregation operator $\mathcal{AGG}_{f_1(A_1),f_2(A_2),...,f_k(A_k)}$ maps each input mapping to a group G and produces one output mapping for each non-empty group G. The output has the form $<G, Val_1, .., Val_k>$, where G is the group identifier (grouping key) and Val_i is the group's aggregate value of the function $f_i(A_i)$. The value Val_i is updated whenever the set of mappings in G changes in the case of new and expired mappings. Both new mappings and expired mappings can result in an update to the value of a group and the aggregate operator needs to report the new value for that group. The incremental computation of each group can be done independently, therefore, they can be assigned to different machines to be computed in parallel. The Global Scheduler can distribute the incremental aggregation tasks by routing new mappings and expired mappings to processing nodes based on their grouping keys. A simple routing policy is splitting based on the hash values of grouping keys. The algorithms for incremental computation of aggregation over sliding windows in [13,10] can be used to incrementally update the value of each group.

4 Implementation and Evaluation

We implemented our elastic execution model and the parallel algorithms using ZooKeeper [16], Storm[1] and HBase.[2] The architecture of CQELS Cloud is shown in Figure 5. The Execution Coordinator coordinates the cluster of OCs using coordination services provided by Storm and HBase which share the same Zookeeper cluster. The Global Scheduler uses Nimbus,[3] an open source EC2/S3-compatible Infrastructure-as-a-Service implementation, to deploy the operators' code to OCs and monitor for failures. Each OC node runs a Storm supervisor which listens for continuous processing tasks assigned to its machine via Nimbus. The processing tasks that need to process the persistent data use the HBase Client component to access data stored in HBase. The machines running an OC also host the HDFS DataNodes of the HBase cluster. The DataNodes are accessed via the OC's HRegionServer component of HBase.

Machines running OCs communicate directly without using intermediate queues via ZeroMQ[4] used inside Supervisors. Their available communication bandwidths are optimized by ZeroMQ's congestion detection mechanism. Based on ZeroMQ, OCs use inter-process communication interfaces as defined by Storm's "spouts" (stream source) and "bolts" (processing) infrastructure by sending tuples. Links among spouts and bolts in a Storm topology indicate how tuples should be passed among them. In CQELS Cloud spouts are used to stream data from sources. Bolts receive any number of input streams from upstream processes that trigger processing pipelines and continually output results as new streams. Processing queries

[1] http://storm-project.net/
[2] http://hbase.apache.org/
[3] http://www.nimbusproject.org/
[4] http://www.zeromq.org/

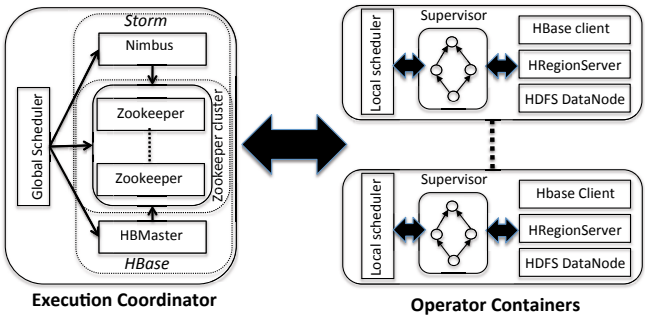


Fig. 5. CQELS Cloud architecture

works as follows: Using the parallelizing algorithms presented in Section 3 a logical query plan is transformed to a Storm topology composed of bolts for all query operators. The stream inputs of the query plan will be mapped to spouts which stream data to special spouts for encoding raw RDF stream. Inner query operators will be connected to the encoding spouts following the acyclic query graph of the logical query plan. Finally, the outermost bolts will be connected to the decoding bolts to transform the query output back into the decoded version.

Data is routed to a bolt using a routing policy, called *stream grouping*. In CQELS Cloud we use three stream grouping policies provided by Storm, namely, *shuffle grouping*, *all grouping* and *fields grouping*. The *shuffle grouping* policy is used to evenly distribute the input to the parallel stateless operator instances. The *all grouping* policy is used to synchronize the input buffers of the parallel multiway join algorithm in Section 3. The *fields grouping* policy is used to route mappings that have a certain key to the aggregation operator instance responsible for computing the aggregated value of the group corresponding to that key (see Section 3.3).

In CQELS Cloud, input mappings are ordered and put into batches carried by Storm tuples that are routed among bolts and spouts. For optimization purposes, we encode the data, thus, mappings contain only fixed-size integers. Consequently, batching a series of mappings in an array of integers will reduce the delay of consuming data from the network as well as the serialization and deserialization. On top of that, this enables us to employs fast compression algorithms such as [9,21] for further speed-up. As the input streams coming to the buffers of an operator instance running in one machine can be unordered, we use the well-established heart-beat approach [24] for guaranteeing the strict order of stream inputs to ensure the correct semantics of the continuous operators.

We use HBase to store the dictionary for the encoding and decoding operations. The dictionary is partitioned by the keys of RDF nodes to store on OC nodes. The encoding and decoding tasks for RDF nodes for the input and output streams are evenly distributed among the nodes using the *fields grouping* policy on hash values of the RDF nodes. HBase stores the written data in memory using its MemStore before flushing partitions of the dictionary into sequential disk blocks on the disks of the destined OC nodes. This allows the writing operations of the dictionary

encoding to be carried out in parallel and has high throughput on OC nodes. Along with the dictionary, HBase is also used to store and index intermediate results from subqueries (which can be huge) on static RDF datasets [19]. This data is stored and indexed by the keys used for the lookup operation which facilitates high throughput of the probing and fetching operations for the parallel processes of the multiway join algorithm.

All state is kept in the Zookeeper cluster which enables high access availability through its built-in replication service. Its reliability, performance and availability can be tuned by increasing the number of machines for the cluster. However, Storm does not directly support state recovery, i.e., resuming a computation state of a node when it crashes. Therefore, we implemented the state recovery for OCs ourselves via timestamped checkpoints which is aided by Storm's concept of guaranteed message processing. Storm guarantees that every input sent will be processed by its *acknowledgment* mechanism. This mechanism allows a downstream processing node to notify its upstream processing node "up to which time point" it has processed downstream inputs successfully. The checkpoint timestamps are encoded in the acknowledgement messages of Storm. This provides an implicit recovery checkpoint when a new node taking over should restart the computation by reconstructing the processing state up to the last "successful" state.

It is important to note that while we use a sophisticated combination of top-class distributed infrastructures, this is "just" the programmatical foundation we base on, in order not to have to deal with classical distributed computing problems ourselves. The mere use of this infrastructures would neither provide elasticity nor scalability of stream processing. Our main contribution lies in the parallel algorithms, operator implementation, and data structures used for this and the highly optimized implementation of these concepts. For example, our operator implementations do not map 1-to-1 to Storm's bolts and spouts, but those are used as a communication components that trigger an execution pipeline from inputs tagged with data workflows.

4.1 Evaluation Setup

To demonstrate the efficiency, performance, scalability and elasticity of our approaches, we evaluated CQELS Cloud using a real deployment on the Amazon EC2 cloud. Our current version of CQELS Cloud uses Zookeeper 3.4.5-cdh4.2, Storm 0.8.2 and HBase 0.94.2. The configuration of the Amazon instances we use for all experiments is "medium" EC2 instances, i.e., 3.5 GB RAM, 1 virtual core with 2 EC2 Compute Units, 410 GB instance storage, 64 Bit platform, moderate I/O performance. A cluster includes 1 Nimbus node, 2 Zookeeper nodes, 1 HBase Master node and 2-32 OC nodes within the same administrative domain. In each experiment, we registered a set of queries or operators and then stream a defined number of stream items to measure the average processing throughput (triples/second or mappings/second).[5] This process is repeated for 2, 4, 8, 16 and

[5] In the following we mean average processing throughput when talking about processing throughput.

32 OC nodes. To test the elasticity, OC nodes are added and removed during the experiments without stopping the CQELS Cloud engine. Our baseline is given by processing the same queries on a single node, i.e., we show how the global scalability can be improved by adding more nodes. This shows the benefits of our central contribution which is the data- and operator-aware load distribution algorithm for stream queries. Our evaluation focuses on showing how multi-joins scale – as in Linked Data star-shaped n-way joins are the dominating operation – when increasing the number of processing nodes, rather than comparing the performance of different join operators. We conducted sets of experiments:[6]

Operator scalability: We evaluate the scalability and overheads of our algorithms proposed in a controlled setting by increasing the number of machines running OCs. For each operator, we fix some parameters to have that kind of processing loads which shows a clear impact of the parallelization. The used data is generated randomly.

Parallel queries: We evaluate the performance and scalability of CQELS Cloud when processing multiple concurrent queries over the Social network scenario of LSBench [20]. LSBench is a benchmarking system for Linked Stream Processing engines which provides a data generator to simulate Social Network streams. We choose LSBench over SRBench [25], because LSBench enables us to control the experimental settings to demonstrate the scalability properties of our system. Furthermore, with LSBench, we can choose four queries with different complexities for our experiments: Q1 (simple matching pattern), Q4 (3-way join on streams only), Q5 (3-way join on stream and static data) and Q10 (3-way join and aggregation). We randomly vary the constant of Q1 to generate 100-100,000 queries and the window sizes of Q4, Q5, Q10 to generate 10-10,000 queries for each. We use LSBench to generate a dataset for 100k users. The dataset presents a social network profile of 121 million triples and 10 billion triples from 5 streams. This dataset can generate tens of millions of intermediate mappings, e.g., more than 33 million for Q5.

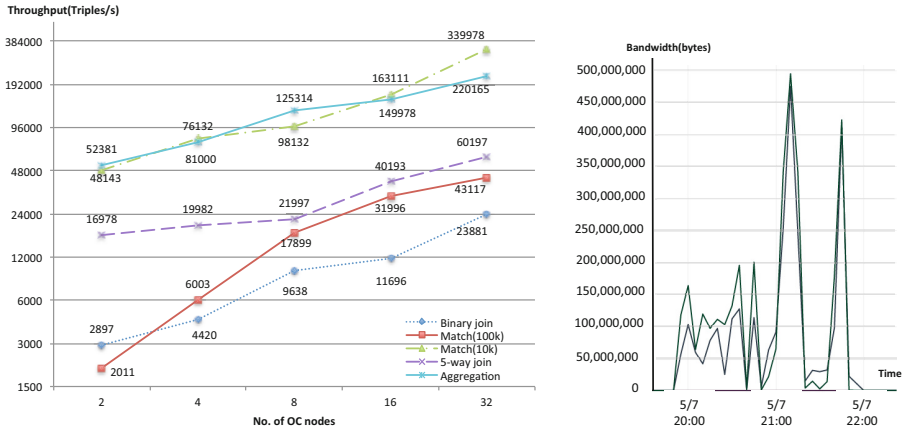
The validity of the throughput measurement for a stream processing engine is defined by the correctness and completeness of its output results [20]. For instance, the mismatches of the outputs generated from different implementations of time-based sliding windows may lead to incorrect interpretations when comparing throughput among them [7]. In our experiments, as we only measure the throughput of one implementation with different configurations, we verify the validity by two types of tests on only count-based windows for any query: unit tests for operator implementations and mismatch tests for output results of queries defined in LSBench [20].

4.2 Evaluation Results

Figure 6a shows the results of 5 experiments: The first two experiments are for the triple matching operator with 10,000 and 100,000 concurrent triple patterns.

[6] A detailed guide for how to reproduce our experiments on Amazon EC2 can be found at `https://code.google.com/p/cqels/wiki/CQELSCloud`

The next two are for the 5-way join and aggregation. The 5-way join has 5 count-based windows of 100,000 mappings and the selectivity of the join predicates is 1%. The aggregation is connected to a window of 1 million mappings. The last one is for a binary join between a window buffer and a bag of 10 million mappings stored in HBase. Note that, the 5-way join and the binary join are two distinct examples to show how the multi-join scales when increasing the number of processing nodes rather than comparing the performance of different join operators.

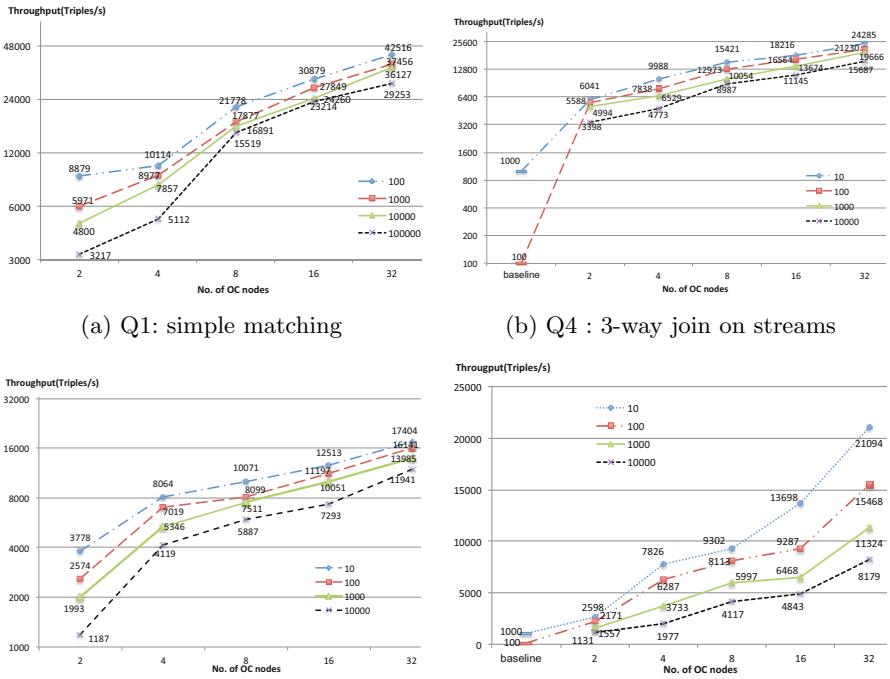


(a) Throughput of Operators (b) Peak network traffic

Fig. 6. Operator behaviors

From the graphs we can see that the throughputs increase linearly with the increasing numbers of OC nodes (logscale on y-axis). This means that the performance increases nearly linearly with the number of processing nodes which is close to the optimal theoretical limit. In terms of throughput, the light-weight operators like triple matching (for 10k triple patterns) and aggregation achieve more than 100k inputs/second with 8 nodes. The ones with heavier query load like triple pattern matching for 100k patterns or the binary join deliver even better scale factors. To confirm that network congestion did not effect the scalability significantly, we show the accumulated network bandwidth of the experiments in Figure 6b. The maximum bandwidth used is less than 500MB/sec for all experiments. This means that we use only a maximum of 4GBit/sec of the 10GBit/sec network offered by the Amazon EC2 cluster and thus the effects incurred by network communication are negligible.

Figure 7 shows more results for four types of queries. Each graph shows the throughputs of an experiment for a certain number of concurrent queries. Similarly to the previous experiments, the throughput of CQELS Cloud increases linearly for a constant query load when we increase the number of OC nodes. When we increase the query load, e.g., by a factor 10, the throughput only decreases



(a) Q1: simple matching

(b) Q4 : 3-way join on streams

(c) Q5: 3-way join on streams and static data

(d) Q10: Join and Aggregation

Fig. 7. Multiple queries on Social Network streams

approximately by a factor of 2. Considering the best throughputs of standalone systems reported in [20] with the same settings for query Q4 and Q10 as baselines, approximately 1000 triples/second for 10 queries and 100 triples/second for 100 queries, the CQELS Cloud can achieve more than 24 times and 210 times of these baseline throughputs for 10 and 100 queries respectively. Furthermore, the more concurrent queries the system handles, the better is the scalability in relation to the baselines. This may be explained by the increasing reuse of intermediate results by multiple queries. This demonstrates excellent scalability and shows the advantages of our data- and query-aware load distribution strategy.

5 Related Work

To the best of our knowledge, CQELS Cloud is the first system addressing elastic and scalable processing for Linked Stream Data but our approaches touch on a number of areas.

Linked Stream Data processing: CQELS Cloud has a data model and query semantics similar to Streaming SPARQL [4], C-SPARQL [3], CQELS [19], etc. However, all these are designed to run on a single machine, while CQELS Cloud

goes beyond that and specifically focuses on scalability issues – discussed in detail in [20] – by distributing the computing and defining an architecture and algorithms suitable for Cloud deployments. There is also preliminary work for distributed stream reasoning on S4 [15] which provides some scalability results for certain queries and reasoning but is not a complete system like ours.

Distributed stream processing engines: Classical distributed stream processing engines such as Borealis [6] and StreamCloud [12] are the distributed versions of the stand-alone engines and only support the relational data model or very generic stream data primitives and operators. They can be used as black-boxes to delegate Linked Data Stream processing tasks, but, as shown in [19,20], the overhead of data transformation and query rewriting seriously impact on scalability, rendering them no competitive option. However, approaches and techniques from distributed stream processing such as load balancing, operator placement and optimizations [11] can be used to improve the performance of CQELS Cloud.

Generic parallel stream computing platforms: Storm and S4 are the most popular elastic stream computing platforms and provide very generic primitives and data types for representing stream elements. None supports declarative query languages nor the Linked Data model. On top of that, neither Storm nor S4 support correlating data from distributed storages as CQELS Cloud does with HBase. There are also other systems such as Kafka[7] and Scribe[8] that target specific, narrow application domains (and do so efficiently). For instance, Kafka and Scribe are used to programmatically create reliable and scalable processing pipelines for stream logs in LinkedIn and Facebook, respectively. We consider these works as complimentary work that we can draw on to potentially improve our implementation.

6 Conclusions

Our goal was to devise scalable algorithms and an infrastructure for Linked Stream processing that scales to realistic scenarios with high stream frequencies, large numbers of concurrent queries and large dynamic and static data sizes along with the possibility to deploy them in a hosted Cloud environment to achieve elasticity in the load profiles and enable "pay-as-you-go" scenarios. The experimental evaluations show that we have achieved this aim to a large degree: Our algorithms and implementation exhibit excellent scalability in the Cloud, essentially supporting arbitrary loads only limited by the number of nodes and the hardware and software characteristics of the used Cloud platform. We achieved this through a completely distributed design with novel parallel algorithms for Linked Stream processing, along with a number of optimization techniques adapted for our purpose and a well-justified combination of sophisticated distributed computing infrastructures. CQELS Cloud provides the same

[7] `http://kafka.apache.org/`
[8] `https://github.com/facebook/scribe`

or even better scalability as the established stream processing approaches outside the Linked Data world and will help to make the Linked Data paradigm an increasingly serious competitor in this area.

References

1. Anicic, D., Fodor, P.: EP-SPARQL: a unified language for event processing and stream reasoning. In: WWW. ACM, New York (2011)
2. Avnur, R., Hellerstein, J.M.: Eddies: continuously adaptive query processing. SIGMOD Rec. 29, 261–272 (2000)
3. Barbieri, D.F., Braga, D., Ceri, S., Grossniklaus, M.: An execution environment for C-SPARQL queries. In: EDBT 2010. ACM, New York (2010)
4. Bolles, A., Grawunder, M., Jacobi, J.: Streaming SPARQL – extending SPARQL to process data streams. In: Bechhofer, S., Hauswirth, M., Hoffmann, J., Koubarakis, M. (eds.) ESWC 2008. LNCS, vol. 5021, pp. 448–462. Springer, Heidelberg (2008)
5. Calbimonte, J.-P., Corcho, O., Gray, A.J.G.: Enabling ontology-based access to streaming data sources. In: Patel-Schneider, P.F., Pan, Y., Hitzler, P., Mika, P., Zhang, L., Pan, J.Z., Horrocks, I., Glimm, B. (eds.) ISWC 2010, Part I. LNCS, vol. 6496, pp. 96–111. Springer, Heidelberg (2010)
6. Daniel, Y.A., Abadi, J.: The Design of the Borealis Stream Processing Engine. In: CIDR 2005, pp. 277–289 (2005)
7. Dell'Aglio, D., Calbimonte, J.-P., Balduini, M., Corcho, O., Della Valle, E.: On correctness in RDF stream processor benchmarking. In: Alani, H., et al. (eds.) ISWC 2013, Part II. LNCS, vol. 8219, pp. 321–336. Springer, Heidelberg (2013)
8. Deshpande, A., Ives, Z., Raman, V.: Adaptive query processing. In: Foundations and Trends in Databases, vol. 1 (January 2007)
9. Elias, P.: Universal codeword sets and representations of the integers. IEEE Trans. Inf. Theor. 21(2), 194–203 (2006)
10. Ghanem, T., Hammad, M., Mokbel, M., Aref, W., Elmagarmid, A.: Incremental Evaluation of Sliding-Window Queries over Data Streams. TKDE 19(1) (2007)
11. Golab, L., Özsu, M.T.: Data Stream Management. Synthesis Lectures on Data Management. Morgan & Claypool Publishers (2010)
12. Gulisano, V., Jimenez-Peris, R., Patino-Martinez, M., Valduriez, P.: Streamcloud: A large scale data streaming system. In: ICDCS (2010)
13. Hammad, M., Aref, W.G., Franklin, M.J., Mokbel, M.F., Elmagarmid, A.K.: Efficient execution of sliding-window queries over data streams. Technical Report 03-035, Purdue University, Dept. of Computer Science (2003)
14. Hammad, M.A., Franklin, M.J., Aref, W.G., Elmagarmid, A.K.: Scheduling for shared window joins over data streams. In: VLDB. VLDB Endowment (2003)
15. Hoeksema, J., Kotoulas, S.: High-performance Distributed Stream Reasoning using S4. In: 1st International Workshop on Ordering and Reasoning, ISWC (2011)
16. Hunt, P., Konar, M., Junqueira, F.P., Reed, B.: Zookeeper: wait-free coordination for internet-scale systems. In: USENIX (2010)
17. Jacobs, A.: The pathologies of big data. Queue 7(6), 10:10–10:19 (2009)
18. Le Phuoc, D.: A Native And Adaptive Approach for Linked Stream Processing. PhD thesis, National University of Ireland, Galway (2013)
19. Le-Phuoc, D., Dao-Tran, M., Xavier Parreira, J., Hauswirth, M.: A native and adaptive approach for unified processing of linked streams and linked data. In: Aroyo, L., Welty, C., Alani, H., Taylor, J., Bernstein, A., Kagal, L., Noy, N., Blomqvist, E. (eds.) ISWC 2011, Part I. LNCS, vol. 7031, pp. 370–388. Springer, Heidelberg (2011)

20. Le-Phuoc, D., Dao-Tran, M., Pham, M.-D., Boncz, P., Eiter, T., Fink, M.: Linked Stream Data Processing Engines: Facts and Figures. In: Cudré-Mauroux, P., Heflin, J., Sirin, E., Tudorache, T., Euzenat, J., Hauswirth, M., Parreira, J.X., Hendler, J., Schreiber, G., Bernstein, A., Blomqvist, E. (eds.) ISWC 2012, Part II. LNCS, vol. 7650, pp. 300–312. Springer, Heidelberg (2012)
21. Lemire, D., Boytsov, L.: Decoding billions of integers per second through vectorization. CoRR, abs/1209.2137 (2012)
22. Manyika, J., Chui, M., Brown, B., Bughin, J., Dobbs, R., Roxburgh, C., Byers, A.H.: Big data: The next frontier for innovation, competition, and productivity. Technical report, McKinsey Global Institute (June 2011)
23. Naughton, V.J.F., Burger, J.: Maximizing the output rate of multi-way join queries over streaming information sources. In: VLDB. VLDB Endowment (2003)
24. Srivastava, U., Widom, J.: Flexible time management in data stream systems. In: ACM SIGMOD-SIGACT-SIGART. ACM, New York (2004)
25. Zhang, Y., Duc, P.M., Corcho, O., Calbimonte, J.-P.: SRBench: A Streaming RDF/SPARQL Benchmark. In: Cudré-Mauroux, P., Heflin, J., Sirin, E., Tudorache, T., Euzenat, J., Hauswirth, M., Parreira, J.X., Hendler, J., Schreiber, G., Bernstein, A., Blomqvist, E. (eds.) ISWC 2012, Part I. LNCS, vol. 7649, pp. 641–657. Springer, Heidelberg (2012)

Towards Constructive Evidence of Data Flow-Oriented Web Service Composition*

Freddy Lécué

IBM Research, Smarter Cities Technology Centre
Damastown Industrial Estate, Dublin, Ireland
firstname.lastname@ie.ibm.com

Abstract. Automation of service composition is one of the most interesting challenges facing the Semantic Web and the Web of services today. Despite approaches which are able to infer a partial order of services, its data flow remains implicit and difficult to be automatically generated. Enhanced with formal representations, the semantic links between output and input parameters of services can be then exploited to infer their data flow. This work addresses the problem of effectively inferring data flow between services based on their representations. To this end, we introduce the non standard Description Logic reasoning *join*, aiming to provide a "constructive evidence" of why services can be connected and how non trivial links (many to many parameters) can be inferred in data flow. The preliminary evaluation provides evidence in favor of our approach regarding the completeness of data flow.

Keywords: Semantic Web, Web Service, Service Composition, Data Flow, Automated Reasoning, Non Standard Reasoning.

1 Introduction

The Semantic Web [1] is considered to be the future of the current Web. In the Semantic Web, Web services [2] are enhanced using rich description languages e.g., OWL the Web Ontology Language [3]. The underlying descriptions, expressed by means of Description Logic (DL) concepts [4] in domain ontologies, are used to describe the semantics of services e.g., their functional inputs, outputs parameters. Intelligent software agents can, then, use these descriptions to reason about Web services and automate their use to accomplish goals specified by the end-user including intelligent tasks e.g., discovery, selection, composition and execution.

We focus on composition and more specially on its *data flow* i.e., links (or connections) which explain how data is exchanged among services (*Right Panel* in Fig.1). While most approaches [5, 6] derive *control flow* of compositions (i.e., a partial order on services - *Left Panel* in Fig.1) according to a goal to achieve, its data flow remains implicit [7] through opaque and pre-defined assignments from incoming to outgoing services. Usually it is up to developers to provide their details e.g., through BPEL (Business Process Execution Language) assign types or filtering/merging operators.

* The research leading to these results has received funding from the European Union's Seventh Framework Programme (FP7/2007-2013) under grant agreement ID 318201 (SIMPLI-CITY).

H. Alani et al. (Eds.): ISWC 2013, Part I, LNCS 8218, pp. 298–313, 2013.

Existing approaches mainly focus in ordering services in a control flow rather than generating its data flow in an automated way. The latter limits flexibility of service oriented computing [8]. Therefore the following are example of open questions in the Web of service community: how to dynamically re-generate data flow specification of "*built-in*" compositions in case of late change of services? Which data is required from which services to turn a composition in its executable state? Does it require data transformation from one description to another? This work investigates the benefits of having semantic descriptions of services a la SA-WSDL [9], OWL-S [10] or WSMO [11] to derive a data flow description of any control flow-based service composition in an automated way.

Towards these issues, some methods [12] exploit expressive DLs to link services through their descriptions, impacting the tractability of the approach. Other approaches [13] limit the expressivity of description through syntactic representation, making data flow very difficult to be automatically derived. In both contexts, complex data links (e.g., filtering, merging) between services cannot be generated in an automated way, providing either abstract or incomplete composition specification. Despite some efforts for pre-defining [14] and inferring [15, 16] compatibilities between services parameters, it remains difficult to derive how data is actually "flowing" from one description to another. In addition, data flow is mainly studied between single outputs and inputs (aka. trivial links). Such links are not appropriate for modeling data flow of complex compositions, limiting their application in real world scenarios. This work tackles this problem.

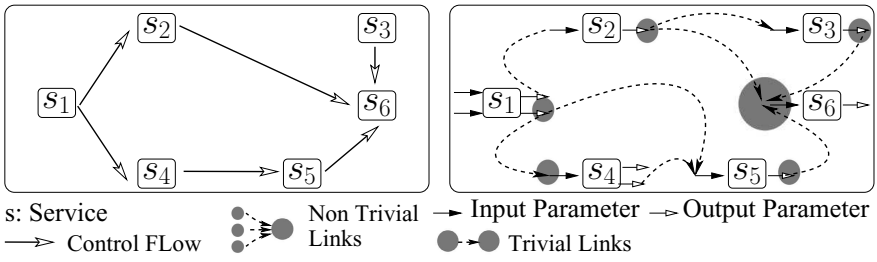


Fig. 1. Control Flow (Left) vs. Data Flow (Right) Views

Suppose some Semantic Web services[1] being organized in a partial order (based on their overall goals): how to effectively infer their non trivial data flow (e.g., filtering, merging). First of all we define non standard DL reasoning *join* to provide a "constructive evidence" of why services can be connected and how non trivial links can be inferred in data flow. The concept *join* is required to exhibit descriptions J from output parameters Out (of services) which properly ensure Out to be compatible with any input parameter In (of services). In other words the description J is constructed for "glue"-ing outputs and inputs parameters of services, and more importantly used for

[1] Polymorphic services (i.e., exposing several functions depending on inputs combinations) are not investigated here, but can be addressed though conditional compositions [6].

understanding how data is flowing among services in a composition. Then we describe how non trivial data flow can be generated, checked and repaired using concept *join* in order to ensure flexible data flow construction. Service descriptions are formalized in $\mathcal{EL}^{++}$, where subsumption and satisfiability are decidable [17]. For the sake of clarity, we assume compositions without open preconditions. Our work assumes that relevant services are already identified and discovered [18]. Control [7] and data flow [19] based composition techniques, combined with the method introduced in the paper, are then applied to derive ready-to-be-executed compositions.

The remainder of this paper is organized as follows. First of all we summarize data flow-oriented composition, its semantic links and limits. Then we present the DL reasoning *join* to provide a "constructive evidence" of why services can be connected. The next sections (i) describe how *join* can be adapted to simulate and construct complex data flow, and (ii) report some experimental results through comparisons with state-of-the-art approaches. Finally we comment on related work and draw some conclusions.

2 Data Flow-Oriented Service Composition

2.1 Service, Semantic Link and Composition

In the Semantic Web, input and output parameters of services are described according to a common ontology or Terminology $\mathcal{T}$ (e.g., Fig.2), where the OWL-S profile, WSMO capability or SA-WSDL can be used as encoding[2], also known as fixed data type or description. Semantic links [19] are defined between output and input parameters of services, based on semantic similarities of their DL encoding. Fig.2 sketches a description of the axioms that are used in the ontology in which the input and output parameters are expressed. Similarities are judged using a matching function between two knowledge representations encoded using the same terminology.

$NetwConnection \equiv \exists netSpeed.Speed$ // *Netw: NetworkConnection*
$Speed \equiv \exists\, mBytes.NoNilSpeed, \quad HighReliable \sqsubseteq Reliable$
$SlowNetwConnection \equiv NetwConnection \sqcap \exists netSpeed.Adsl1M$
$USProvider \equiv \exists to.US,\ UKProvider \equiv \exists to.UK,\ UK \sqcap US \sqsubseteq \bot$
$EUProvider \equiv \exists to.EU, UK \sqsubseteq EU, EU \sqcap US \sqsubseteq \bot, Business \sqsubseteq \top$
$Adsl1M \equiv Speed \sqcap \exists\, mBytes.1M, \quad 1M \sqsubseteq NoNilSpeed$

Fig. 2. DL $\mathcal{EL}^{++}$ Axioms used for representing Output and Input Parameters

In this context, data flow-oriented service composition consists in retrieving semantic links $sl_{i,j}$:

$$sl_{i,j} \doteq \langle s_i, Sim_{\mathcal{T}}(Out, In), s_j \rangle \tag{1}$$

between an output parameter Out of service s_i and input parameter In of service s_j, where both Out and In are DL descriptions. Thereby s_i and s_j are partially linked

[2] In case of multiple ontologies used for services descriptions, alignment techniques [20] need to be investigated.

according to a matching function $Sim_{\mathcal{T}}$, specifying its data flow. Given a terminology $\mathcal{T}$, the range of $Sim_{\mathcal{T}}$ is determined by five matching types following [21, 22]: i) *Exact* i.e., $Out \equiv In$, ii) *PlugIn* i.e., $Out \sqsubseteq In$, iii) *Subsume* i.e., $In \sqsubseteq Out$, iv) *Intersection* i.e., $\neg(Out \sqcap In \sqsubseteq \bot)$ and v) *Disjoint* i.e., $Out \sqcap In \sqsubseteq \bot$. The cases i)-iv) identify compatible descriptions while the case v) identifies incompatible descriptions Out and In.

2.2 Limitations

As stated in Introduction, models such as (1) are mainly considered for representing trivial semantic links i.e., (boolean) one-to-one compatibility (though matching types) between single output and input parameters. Towards this issue, we generalize (1) by considering In and Out respectively as a conjunction of inputs and outputs of services. Semantic links between "any" output and input at a time i.e., non trivial data flow, can be then represented in (1), which is more appropriate for modeling complex data flow.

However such a model is still limited to understand how data is "flowing" from services to services. Indeed, how data is properly manipulated and adapted between services to ensure data flow? Which part of services descriptions is the most relevant? Is it maximal, minimal, effective and how? These are general questions which remain open in the join domains of Semantic Web and Web of services.

This work suggests concept join as a constructive reasoning to provide a "constructive evidence" of why services can be connected and how complex data flow can be inferred in services composition.

3 Towards Constructive Evidence of Data Flow

Towards the issue of explaining why services can be connected and how non trivial links can be inferred in data flow, Section 3.1 introduces the innovative concept join (Definitions 1, 2 and Propositions 1,2) between data descriptions. Section 3.2 follows the methodology of [23] and [24] to prove the computational complexity of the join reasoning. In particular Proposition 3 is inspired from [23], but highly adapted to concept join (which constructs different descriptions - see Section 6.2). Section 3.3 combines in an innovative way state-of-the-art abduction (Definition 3) and contraction (Definition 4) reasoning techniques to extend the applicability of concept join in a (i) context of service composition, and (ii) when Proposition 1 does not hold (Algorithm 2). Importantly, Section 3.3 explains how non standard reasoning abduction and contraction can be used for enriching the number of joins between services in a composition.

3.1 Concept Join: Definitions and Propositions

We are interested in descriptions in Out which ensure Out and In to be compatible. Therefore we aim at extracting J (Join - Definition 1) from Out such that $J \sqsubseteq In$ remains true in $\mathcal{T}$ (Definition 1). The descriptions R (Remainder), part of Out, such that $Out \equiv R \sqcap J$ will need to be removed from Out since they move Out away from In under subsumption $\sqsubseteq_{\mathcal{T}}$. J highlights descriptions which could be properly joined

with In in order to compose outputs Out and inputs In while R points out descriptions which are not required by In.

Definition 1 ***(Concept Join)***
Let $\mathcal{L}$ be a DL, Out, In be two concepts in $\mathcal{L}$, and $\mathcal{T}$ be a set of axioms in $\mathcal{L}$ such that $\mathcal{T} \not\models Out \sqcap In \sqsubseteq \bot$. A Concept Join Problem, denoted as $CJP\langle \mathcal{L}, Out, In, \mathcal{T}\rangle$ (shortly $Out \blacktriangleright In$) is finding a pair of concepts $\langle R, J\rangle \in \mathcal{L} \times \mathcal{L}$ such that i) $\mathcal{T} \models Out \equiv R \sqcap J$ and ii) $\mathcal{T} \models J \sqsubseteq In$. Then J (or $\blacktriangleright_J$), which is not symmetric, is a join between Out and In in $\mathcal{T}$.

We use $\mathcal{P}$ as a symbol for a $CJP\ \langle \mathcal{L}, Out, In, \mathcal{T}\rangle$ and we denote with $SOLCJP(\mathcal{P})$ the set of all solutions of the form $\langle R, J\rangle$ to a $CJP\ \mathcal{P}$. In case $\mathcal{T} \not\models Out \sqsubseteq In$, the $CJP\ \mathcal{P}$ has no solution at all, as stated formally in Proposition 1.

Proposition 1. ***(No Solution of a CJP)***
Let $\mathcal{P} = \langle \mathcal{L}, Out, In, \mathcal{T}\rangle$ be a CJP such that $\mathcal{T} \not\models Out \sqsubseteq In$. The set $SOLCJP(\mathcal{P})$ is defined by $\emptyset$.

Proof. Since Out can be rewritten as $R \sqcap J$ (condition (i) in Definition 1) with $R = \top$ and $J = Out$ without loss of generality, then $\mathcal{T} \not\models Out \sqsubseteq In$ (Proposition 1) becomes $\mathcal{T} \not\models J \sqsubseteq In$. The latter contradicts $\mathcal{T} \models J \sqsubseteq In$ (condition (ii) in Definition 1), so no possible solution of a $CJP\ \mathcal{P}$ in case $\mathcal{T} \not\models Out \sqsubseteq In$.

$\mathcal{T} \models Out \sqsubseteq In$ implies that there is always the trivial solution $\langle \top, Out\rangle$ to a CJP $\langle \mathcal{L}, Out, In, \mathcal{T}\rangle$.

Proposition 2. ***(Trivial Solution of a CJP)***
If $Out \equiv In$ in $\mathcal{T}$ then $\langle \top, Out\rangle \in SOLCJP(\langle \mathcal{L}, Out, In, \mathcal{T}\rangle)$.

This case refers to an *exact* composition [25] of services s_i and s_j: if we want to proceed s_j, all outputs Out of s_i are required (since J is defined by Out in Proposition 2) to achieve all input In of s_j. Then, no description R has to be removed from Out. On the other hand, when $Out \sqsubset In$ (i.e., $\mathcal{T} \models Out \sqsubseteq In$ and $\mathcal{T} \not\models Out \equiv In$), $\langle \top, Out\rangle$ is also one potential solution of the CJP problem. However, other solutions with R not being $\top$ are possible. Obviously, in order to achieve a composition between Out and In the first case (in Proposition 2) is in a much better shape than the second one. Indeed all descriptions In, which are required by s_j, are provided by Out. If we want to use join to highlight the closest descriptions in Out (i.e., the most general) to In, emphasising the most compatible descriptions in Out for In to compose s_i and s_j, "effective" joins under $\sqsubseteq_{\mathcal{T}}$ need to be defined (Definition 2 adapted from [26]).

Definition 2 ***(Effective Join Solution)***
Let $\mathcal{P} = \langle \mathcal{L}, Out, In, \mathcal{T}\rangle$ be a CJP. The set $SOLCJP_{\sqsubseteq}(\mathcal{P})$ is the subset of $SOLCJP(\mathcal{P})$ whose join concepts J are maximal under $\sqsubseteq_{\mathcal{T}}$. The set $SOLCJP_{\leq}(\mathcal{P})$ is the subset of $SOLCJP(\mathcal{P})$ whose join concepts have minimum length.

Formally the set $SOLCJP_{\sqsubseteq}(\mathcal{P})$ satisfies both Definition 1 and the following condition: $\forall \langle R', J'\rangle \in \mathcal{L} \times \mathcal{L} : \mathcal{T} \models Out \equiv R' \sqcap J' \wedge \mathcal{T} \models J' \sqsubseteq In \Rightarrow J' \sqsubseteq J$. Maximality under $\sqsubseteq_{\mathcal{T}}$ is considered as a effectiveness criterion since no unnecessary joins is assumed between Out and In.

Example 1 ***(Effective Join Solution - Fig.3)***
Let s_1 be an `InternetEligibility` *service which returns as output Out: the NetworkConnection (e.g., Speed, UK Country) of a desired geographic zone together with information about its network provider (Reliability, Business type). Let s_2 be another telecom service which requires a Reliable network provider in UK as input In to be executed. Out and In, as DL representations of functional parameters in Fig.3, ensure $Out \sqsubseteq In$ in $\mathcal{T}$. On the one hand $\exists netSpeed.Adsl1M \sqcap \exists to.UK \sqsubseteq NetwConnection$. On the other hand $HighReliable \sqsubseteq Reliable$. In other words some outputs produced by s_1 can be consumed by some inputs of s_2. The effective join J of Out and In (under $\sqsubseteq_{\mathcal{T}}$) is $\exists netSpeed.Adsl1M \sqcap \exists to.UK \sqcap HighReliable$ while the discarded description R is $Business$. An instance of J is then required to instantiate In (and execute s_2): $SlowNC$ (NC refers to $NetwConnection$), $\exists to.UK$, $Reliable$ while an instance of $Business$ is not. The description J acts as a filter between s_1, s_2 to restrict Out over the data flow. In other words J establishes which descriptions are relevant to link Out to In. The two output instances of s_1 are then practically merged into one instance for s_2 through the construction of J.The latter ensures the executability of s_2.*

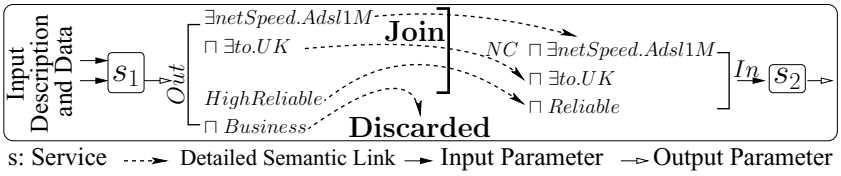


Fig. 3. Effective Join Solution

In [26] it was proven that $\leq$-minimality is more appropriate for conciseness, but largely depending on $\mathcal{T}$. Indeed, by simply adding axioms $A \equiv R$ and $B \equiv J$, we obtain a $\leq$-minimal solution $\langle A, B\rangle$ for each pair $\langle R, J\rangle \in SOLCJP(\mathcal{P})$.

3.2 Computational Complexity

Since concept join can be considered as an extension of concept subsumption with respect to a TBox, its lower bounds carry over to decision problems related to a CJP.

Proposition 3. ***(Deciding Existence of Join)***
Let $\mathcal{P} = \langle \mathcal{L}, Out, In, \mathcal{T}\rangle$ be a CJP. If concept subsumption with respect to a $\mathcal{T}$ in $\mathcal{L}$ is a problem $\mathcal{C}$-hard for a complexity class $\mathcal{C}$, then deciding whether a pair of concepts $\langle R, J\rangle \in \mathcal{L} \times \mathcal{L}$ belongs to $SOLCJP(\mathcal{P})$ is $\mathcal{C}$-hard.

Proof. Since $\mathcal{T} \models Out \sqsubseteq In$ iff $\langle \top, Out\rangle \in SOLCJP(\mathcal{P})$, such a problem is $\mathcal{C}$-hard.

In our $\mathcal{EL}^{++}$ context, deciding whether a pair of concepts $\langle R, J\rangle$ belongs to $SOLCJP(\mathcal{P})$ is PTIME-hard [27] with respect to both acyclic and cyclic TBoxes $\mathcal{T}$.

Regarding upper bounds, a simple result can be derived from the fact that $\langle \top, Out\rangle$ is always a solution of the CJP $\langle \mathcal{L}, Out, In, \mathcal{T}\rangle$ if $Out \sqsubseteq In$ in $\mathcal{T}$ (Proposition 2) although not always an effective one for join. Following [23], a total length-lexicographic

order $\prec_{lex}$ can be defined over concepts as follows: given two concepts $Out, In \in \mathcal{L}$, let $Out \prec_{lex} In$ if either $|Out| < |In|$, or both $|Out| = |In|$ and Out is lexicographically before In. Based on this total order, an approach for finding a $\leq$-minimal solution of a CJP, using polynomial space relatively to an oracle for subsumption in $\mathcal{L}$, is presented in Algorithm 1. Algorithm 1 is innovative as it enumerates concept join solutions over a total length-lexicographic ordered concepts.

Algorithm 1. Effective $\blacktriangleright_J$ of a CJP

```
1 Input: A CJP P = ⟨L, Out, In, T⟩ with T ⊨ Out ⊑ In.
2 Result: A concept x ∈ L such that ⟨R, x⟩ ∈ L × L is in SOLCJP≤(P).
3 begin
4     x ← ⊤; // Initialisation
5     while |x| < |Out| do
6         if T ⊨ Out ⊏ x and T ⊨ x ⊑ In then
7             return x;
8         x ← next concept following x in ≺lex;
9     x ← Out; return Out;
```

Algorithm 1 uses polynomial space (considering one call to subsumption as an oracle) since it just tries all concepts with less symbols than Out, and returns Out if it does not find a shorter solution. Thus, it provides an upper bound on the complexity of CJP, depending on the complexity class to which subsumption in $\mathcal{L}$ belongs to. Although this result does not directly lead to a practical algorithm, it provides an upper bound on the complexity of the problem, hence on the complexity of every optimal algorithm.

Theorem 1. ***(Finding a Solution in*** $SOLCJP_{\leq}(\mathcal{P})$***)***
Let $\mathcal{P} = \langle \mathcal{L}, Out, In, \mathcal{T} \rangle$ be a CJP. If concept subsumption with respect to a $\mathcal{T}$ in $\mathcal{L}$ belongs to a complexity class $\mathcal{C}$ that is included in PSPACE then finding a pair of concept in $SOLCJP_{\leq}(\mathcal{P})$ is a problem in PSPACE. Otherwise if PSPACE is included in $\mathcal{C}$, then finding a pair of concept in $SOLCJP_{\leq}(\mathcal{P})$ is a problem in $\mathcal{C}$.

According to Theorem 1, inspired from [26], finding a pair of concept for the problem $SOLCJP_{\sqsubseteq}(P)$ in $\mathcal{EL}^{++}$ is in PSPACE. Theorem 1 simply builds on top of the subsumption properties.

3.3 Incompatible Descriptions in Concept Join

As highlighted by Proposition 1, Definition 1 has no solution if $\mathcal{T} \not\models Out \sqsubseteq In$. This limits the applicability of concept join by restricting services to exchange data (from Out to In) only under $Out \sqsubseteq In$ in $\mathcal{T}$. Even if this is a basic requirement to compose and join services, other potential compositions, which do not satisfy $Out \sqsubseteq In$ [25], would be ignored since their join cannot be derived. Towards this issue, we exploit constructive DL reasoning abduction [28] (Definition 3) and contraction [24] (Definition 4) to respectively consider join if i) In does not subsume Out but have a consistent conjunction i.e., $\mathcal{T} \not\models Out \sqcap In \sqsubseteq \bot$ and ii) their conjunction is inconsistent

i.e., $\mathcal{T} \models Out \sqcap In \sqsubseteq \bot$. While concept abduction derives description which is missing in Out to be subsumed by In, concept contraction [24] retracts specification G (for *Give up*) in Out to obtain a concept K (for *Keep*) such that $K \sqcap In$ is satisfiable in $\mathcal{T}$. The latter extends abduction to unsatisfiable conjunction of Out and In.

Definition 3 ***(Concept Abduction)***
Let $\mathcal{L}$ be a DL, Out, In be two concepts in $\mathcal{L}$, and $\mathcal{T}$ be a set of axioms in $\mathcal{L}$ such that $\mathcal{T} \not\models Out \sqcap In \sqsubseteq \bot$. A Concept Abduction Problem: $In \backslash Out$ is finding a concept $H \in \mathcal{L}$ such that $\mathcal{T} \not\models Out \sqcap H \equiv \bot$, and $\mathcal{T} \models Out \sqcap H \sqsubseteq In$.

Similarly to concept join, abduction extends subsumption. It also constructs a concept H to ensure $Out \sqcap H$ be subsumed by In. By computing description H using abduction, join can be derived between $Out \sqcap H$ (instead of Out) and In. Abduction is then required to enlarge the scope of Definition 1 i.e., from $Out \sqsubseteq In$ to $\neg(Out \sqcap In \sqsubseteq \bot)$ in $\mathcal{T}$.

Contraction, which extends satisfiability, aims to retract specification G (for *Give up*) in Out to obtain a concept K (for *Keep*) such that $K \sqcap In$ is satisfiable in $\mathcal{T}$.

Definition 4 ***(Concept Contraction)***
Let $\mathcal{L}$ be a DL, Out, In be two concepts in $\mathcal{L}$, and $\mathcal{T}$ be a set of axioms in $\mathcal{L}$ where both Out and In are satisfiable in $\mathcal{T}$. A Concept Contraction Problem, denoted as $In \lrcorner Out$ is finding a pair of concepts $\langle G, K \rangle \in \mathcal{L} \times \mathcal{L}$ such that $\mathcal{T} \models Out \equiv G \sqcap K$ and $\mathcal{T} \not\models K \sqcap In \sqsubseteq \bot$. Then K (or $\lrcorner_K$) is a contraction of Out according to In and $\mathcal{T}$.

By computing (1) contraction $\lrcorner_K$: a part of Out which ensures $\lrcorner_K \sqcap In$ to be satisfiable in $\mathcal{T}$ (i.e., validating conditions of Definition 3), and then (2) abduction $In \backslash \lrcorner_K$ which ensures $\lrcorner_K \sqcap (In \backslash \lrcorner_K) \sqsubseteq In$, join can be derived between $\lrcorner_K \sqcap (In \backslash \lrcorner_K)$ and In. Thus contraction can be applied to enlarge the scope of Definition 1: from $Out \sqsubseteq In$ to $Out \sqcap In \sqsubseteq \bot$ in $\mathcal{T}$.

Algorithm 2 sketches the approach to enlarge the scope of Definition 1. It ensures that Out and In can be joined by iteratively weakening and strengthening Out through contraction and abduction. Besides the case already supported by Propositions 1 and 2 and its extension to $Out \sqsubset In$ (line 6), abduction (lines 10, 14) is applied if $Out \sqcap In$ is consistent (line 9) in $\mathcal{T}$. Alternatively contraction (line 13) is required beforehand (line 12). The most specific contraction is considered to obtain a description as close as possible to Out. Thus, the join is derived between (1) Out and In in the trivial case $Out \sqsubseteq In$ (line 6), (2) $Out \sqcap (In \backslash Out)$ and In if $\mathcal{T} \not\models Out \sqcap In \sqsubseteq \bot$ (line 9) and (3) $(In \lrcorner_K Out) \sqcap (In \backslash (In \lrcorner_K Out))$ and In if $\mathcal{T} \models Out \sqcap In \sqsubseteq \bot$ (line 12).

The complexity of Algorithm 2 is in PSPACE in $\mathcal{EL}^{++}$. Indeed lines 6, 9, 12 are in PTIME [17], line 13 is in PTIME (Theorem 4 in [24]), lines 10, 14 are in PSPACE (Theorem 1 in [28]), line 15 is in PSPACE (Theorem 1).

4 Composing Services with Concept Join

We present how concept join can be used to compose properly services through complex data flow modelling.

4.1 Join-ing Data and Descriptions of Services

Compositions of any outputs Out with inputs In can be derived using Algorithm 2. The data flow is established by joining their descriptions. In case their join cannot be derived (lines 9 and 12), we apply contraction and abduction to identify data descriptions which need to be removed/added from/to outputs Out of services with respect to inputs In.

Algorithm 2. Computing Join (Case $\mathcal{T} \not\models Out \sqsubseteq In$)

1 **Input**: A $CJP\ \mathcal{P} = \langle \mathcal{L}, Out, In, \mathcal{T} \rangle$.
2 **Result**: A pair $\langle R, J \rangle \in \mathcal{L} \times \mathcal{L}$ which is in $SOLCJP_{\sqsubseteq}(\mathcal{P})$.
3 **begin**
4 $H \leftarrow \top$; *//Initialisation*
5 *// Trivial Case of Subsumption between Out and In.*
6 **if** $\mathcal{T} \models Out \sqsubseteq In$ **then**
7 ; *// Propositions 1, 2 and its Extension to $Out \sqsubset In$.*
8 *// Extension to Consistent Conjunction | $\mathcal{T} \not\models Out \sqcap In \sqsubseteq \bot$.*
9 **else if** $\mathcal{T} \not\models Out \sqcap In \sqsubseteq \bot$ **then**
10 $H \leftarrow In \backslash Out$; *// Abduction*
11 *// Extension to Inconsistent Conjunction of Out and In.*
12 **else if** $\mathcal{T} \models Out \sqcap In \sqsubseteq \bot$ **then**
13 $Out \leftarrow (In \lrcorner_K Out)$; *// Contraction*
14 $H \leftarrow In \backslash Out$; *// Abduction*
15 $\langle R, J \rangle \leftarrow SOLCJP_{\sqsubseteq} \langle \mathcal{L}, Out \sqcap H, In, \mathcal{T} \rangle$; *// Min. Join*
16 **return** $\langle R, J \rangle$;

In some cases, Semantic Web services *consumed* and *produced* data that does not fit its static semantic description, making semantics of data not as precise as it should be. In this context, we proceed as following: (1) detecting the most accurate semantic description of concrete data values following [20]), (2) expanding the domain ontology with this new description, mainly for reasoning purpose, and (3) applying Algorithm 2 at run time to obtain joins. The steps (1) and (2) ensures that the reasoning at description level (through Algorithm 2) is also valid at a lower (i.e., data) level. This case of non-alignment between data and their description justifies and reinforces the use of non standard reasoning to capture composition. Indeed, more inconsistent joins could occur, limiting the applicability of pure equivalence-based approaches [16].

4.2 Simulating Complex Data Flow Operators

Definition 1, as a way to identify (semantic) link-"*able*" descriptions in composition, can be used to simulate/infer complex data flow operators e.g., "*Data Filter*", "*Merge*". Their benefit is twofold: modeling and explaining how services and their data can be properly manipulated and adapted in data flow-oriented composition. Contrary to [25, 16, 6], among others, automated generation, verification and repair of complex data

flow in composition can be enabled once integrated in a composition engine [29]. In the following the symbol ▶ will denote the problem in Definition 1 where both (i) effective join solutions (Definition 2) and (ii) maximality under $\sqsubseteq_{\mathcal{T}}$ are considered.

• **Data Filter**: [14] commonly used the data filter operator in data flow-oriented service composition to i) extract some descriptions Y and ii) block the rest In from an incoming description X with respect to a filter (description) Z (see illustration in Fig.4). This operator is simulated by $X \blacktriangleright Z$ and its solution $\langle In, Y \rangle$. $X \sqsubseteq Z$ since Z is used as a filter for X. The effectiveness condition (Definition 2) is crucial to avoid any undesired data in Y e.g., In. The more specific the filter Z (i.e., the closer to X), the less descriptions blocked by Z (the least is $\top$).

Example 2 ***(Data Filter - Fig.4 a))***
Let Y be defined by $\exists to.UK$ and D be defined by $Business$. The descriptions Y and D are respectively extracted and blocked from description X i.e., $\exists to.UK \sqcap Business$ using the filter Z, defined by $\exists to.EU$. Each data instance from X is split along Y and D. Only instance of X is connected to Y.

• **Data Merge**: In [7] it is used to aggregate descriptions X_1 and X_2 into a description Y with respect to a filter Z (see illustration in Fig.4). If X_1 and X_2 are compatible, this operator can be simulated by $(X_1 \sqcap X_2) \blacktriangleright Z$ and its solution $\langle In, Y \rangle$. $X_1 \sqcap X_2 \sqsubseteq Z$ since Z is used as a filter for X_1 and X_2. In refers to descriptions which are blocked from X_1 and X_2 with respect to Z. In case $X_1 \sqcap X_2 \equiv Z$, all descriptions from $X_1 \sqcap X_2$ are merged, ensuring In to be $\top$ i.e., none of descriptions in $X_1 \sqcap X_2$ is blocked from Y. A generalization to n descriptions to merge is straightforward.

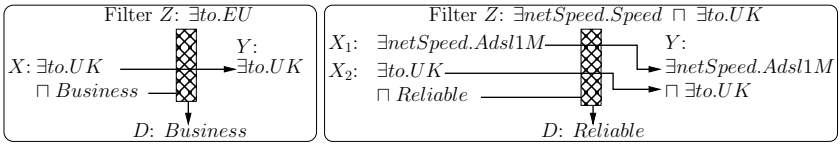


Fig. 4. Simulation of Data Filter and Merge with Join

Example 3 ***(Data Merge - Fig.4 b))***
$\exists netSpeed.Adsl1M \sqcap \exists to.UK$ is the merging description of X_1, X_2 in Fig.4 b) using the filter $\exists netSpeed.Speed \sqcap \exists to.UK$ while $Reliable$ is the description which is blocked.

Based on a straightforward extension of Algorithm 2 with effective concept join, most common complex data flow operators e.g., *Data Merge*, *Filter* can be derived in any data flow, modeling and explaining how services and their data are adapted. Algorithm 2 can be also used to validate pre-defined links or complete existing ones. More generally effective concept join can be used in any data-based application e.g., as a way to retrieve instances of Z from a large set of data Y given some constraints X i.e., $Y \blacktriangleright Z$.

5 Experimental Results

In more details we analyze our approach (Algorithm 2 and its extension for data flow simulation) by comparing its performance against existing approaches [5–7] along two dimensions: (i) CPU time (in ms) to generate composition and (ii) completeness of data flow. The second dimension is evaluated by computing the rate: data descriptions connections retrieved against those expected in the optimal composition. This composition, which is manually constructed based on services descriptions and their goal, has no open links (i.e., links reaching to a non executable process) and no redundant links. The experiments have been conducted on Intel(R) Core (TM)2 CPU, 2.4GHz, 2GB RAM.

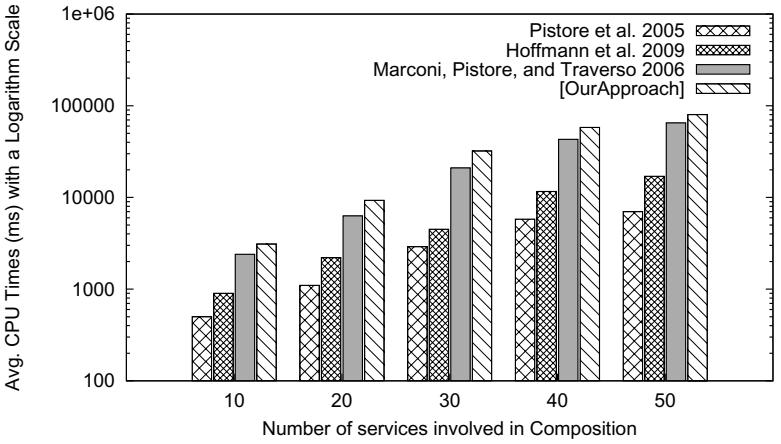


Fig. 5. Computation Time of Composition Approaches

• **Context:** Compositions with up to 50 services have been extracted from [30] and enriched using a commercial $\mathcal{EL}^{++}$ ontology (1100 concepts, 390 properties: 384 concepts subsume the 716 remaining ones with a maximal depth of 8). The semantic annotations are important for deriving data flow in our approach. SOUR[3] is used for services annotations. The annotation process is costly e.g., 8 person/hours for 50 services (with an average of 5 inputs/outputs) with the latter ontology, but has a positive impact on automation of compositions. For scalability purpose we guided the semantic link detection since each composition is bound by $n \times 2^n$ potential semantic links, with n be the number of services. In more details we limited the number of Out (input of Algorithm 2) to be computed beforehand e.g., by ranking Out with respect to In (e.g., size of their contraction/abduction) and considering only Out which ensures to obtain the top k contraction/abduction. The semantic link detection was required only by our approach, mainly to (i) identify potential data flow in composition and (ii) avoid the computation of an exponential number of join, which strongly reduce the overall computation time. The data flow requirements are formalized for [7] while only composition goals are defined for [5, 6].

[3] `http://www.soa4all.eu/tools.html`

• **Results - Computation Time:** Fig.5 illustrates the computation costs for constructing compositions with up to 50 services. Our approach is the most time consuming although (i) a control flow-based compositions is pre-defined and (ii) conjunctions of outputs are considered satisfiable. Other approaches, generating control flow-based compositions, are faster. The best approach [5] generates compositions of 50 services in 7.2 seconds.

• **Results - Data Flow Completeness:** Fig.6 sketches the comparison of our approach vs. existing approaches. The same number of compositions has been retrieved in all cases. The only difference is related to its data flow description. On average our approach automatically derives 83% of the final data flow structure (i.e., data filter, merge operators) of a data flow-free composition. The 17% remaining connections, are cyclic-based data flow operators e.g., loop, which is not supported by our current implementation. On average no more than 55% of connections are retrieved with the state-of-the-art approach [7]. The approach of [5] generates an average of 9% of connections. As reported by their authors, this is more appropriate for independent services.

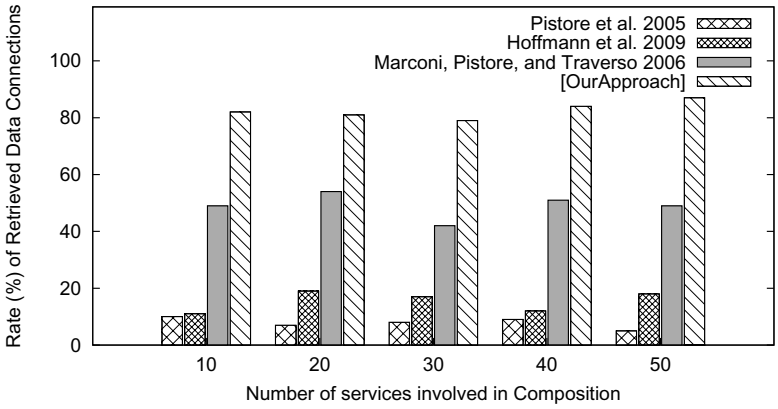


Fig. 6. Data Flow Completeness

• **Lessons Learned:** Even if state-of-the art approaches are appropriate for fast elaboration of control-flow-based composition, they are not necessarily adequate for (i) detecting connections between services and (ii) connecting their descriptions. The automated construction of complex data flow in $\mathcal{EL}^{++}$ DL has a negative impact on the computation costs but ensures a finer description of compositions, which are ready for execution. The size and the structure of the ontology have a limited impact. The main factors for the increase of computation cost are (i) the expressivity of the DL and (ii) the number of DL conjuncts (and their complexity) used to describe services. The reduction of its expressivity has a positive impact on scalability, but it also decreases the completeness and quality of data flow. The scalability can be improved by considering only subsumption-based comparisons of descriptions (line 6), removing computation of abduction and contraction. In such a case the rate of data flow completeness is

also decreasing. By removing the abduction and contraction parts of Algorithm 2 (from line 9 to 14), our approach is more scalable than state-of-the-art approaches, but only 55% of data flow description is retrieved. According to our experiments a best trade-off is proposed in [7], while [5, 6] fits perfectly independent services with a better scalability for [5].

• **Limitations:** The computed potential connections are all used for defining the data flow of the composition. However if multiple services provide similar output (respectively input) descriptions, they are all equally considered. All their output (respectively input) descriptions are aggregated and subject to a join with other services. This case falls in a special case of "Data Merge" (Fig.4 b where $X_1 \equiv X_2$, with X_1 and X_2 outputs of two distinct services). Additional manual efforts are required if such cases need to be avoided, which were not foreseen in our applications.

6 Related Work

6.1 Data Flow-Based Semantic Service Composition

Fig.7 positions existing approaches in relation to 3 dimensions: control flow, data flow, description expressivity. These dimensions aree used to structure the remainder of this section.

Mash-up-based approaches [31, 13] and semantics-based methods [7, 32, 14], positioned in *Front Cluster* of Fig.7, achieve composition by linking services according to different expressivity of static control flow and pre-defined data flow operators (with explicit requirements). They are all limited by the expressivity of service descriptions. Indeed the latter are constrained by RDF/S while the former support only basic XML-based transformation. By embedding compositions with advanced control flow [7], the data flow construction is reduced. [14] provide a more complete (pre-designed) panel of data flow operators, such as *Construct* and *Mix*, which can be simulated by Definition 1, but support only RDF/S, focusing at instance level. Their applicability to expressive semantics and the automated construction of data flow is then limited.

AI planning- [6, 33] and DL-based approaches [15, 12], positioned in *Back Cluster* of Fig.7) elaborate composition of services by reasoning on their descriptions. Despite higher expressivity, only sequence-based data flow is inferred. The approaches of [15, 25, 32] are even more restrictive as they consider (specialized) semantic links between one output and input. More elaborated operators have been presented by [16] towards this issue. Contrary to our approach, data flow is based on concrete values and not their semantic descriptions, which is more flexible for handling misalignment data-description e.g., the instance defined by $(\exists hasConnection.ADSL512KBS)$ where $ADSL512KBS$ is a $SlowNetwConnection$ partially respects the description $SlowNetwConnection \sqcap \exists to.UK$. Indeed no instance of a provider is provided. We address it by using non standard reasoning. Other approaches simulate sequence [33] and conditional-based [6], e.g., through forward effects for the latter, limiting the expressivity of compositions.

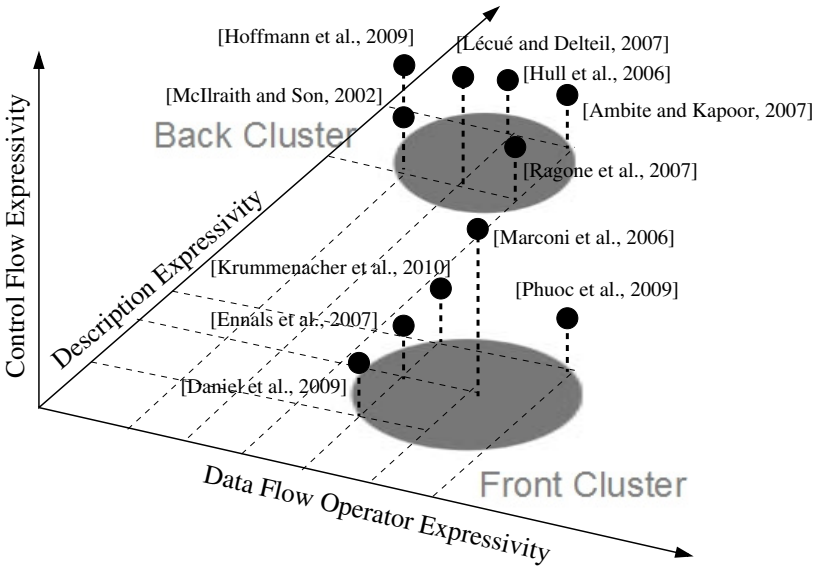


Fig. 7. Classification of Data Flow-oriented Composition

6.2 Existing Constructive DL Reasoning

While abduction [26] derives description which is missing in Out to be subsumed by In, concept contraction [24] retracts specification G (for *Give up*) in Out to obtain a concept K (for *Keep*) such that $K \sqcap In$ is satisfiable in $\mathcal{T}$. The latter extends abduction to unsatisfiable conjunction of Out and In. Approximate subsumption has been presented by [34]. Such types of reasoning construct concepts which are missing or over-specified in Out to be respectively (1) subsumed by and (2) consistent with In. Concept join constructs more general concepts from Out which are subsumed by In. In particular, its effective solutions (under $\sqsubseteq_{\mathcal{T}}$) refer to the most general description of Out which is subsumed by In. Abduction and approximate subsumption extend Out while join extracts a part of Out for the same objective i.e., being subsumed by In. If $Out \sqsubseteq In$, abduction, contraction and approximate subsumption do not construct any description while concept join does. It explains the way they are joined.

Subsumption between DLs concepts Out and In can be explained by deriving its formal proof (i.e., which descriptions in In subsume which descriptions in Out) in [35]. Concept join does not provide any explanation of subsumption, but instead closer descriptions J (in Out) of In given Out under $\sqsubseteq_{\mathcal{T}}$.

7 Conclusion

In this paper we studied data flow-oriented Web service composition. Our work has been directed to meet the main challenges facing this problem i.e., *how to effectively*

infer data flow between services based on their DL $\mathcal{EL}^{++}$ descriptions? Firstly we introduced the constructive reasoning *join* in $\mathcal{EL}^{++}$, aiming to provide a "constructive evidence" of why services can be connected. Then we described how non trivial data flow can be generated, checked and (potentially) repaired using concept *join*, all ensuring flexible data flow construction. Thus, implications of control flow modification on data flow can be investigated. The experimental results provide evidence in favor of our approach regarding the completeness of data flow.

Future works will focus on modeling data flow operators at instance level [14] i.e., how do loops in control flow work together with data flow? We will also investigate metrics for evaluating data flow precision.

References

1. Berners-Lee, T., Hendler, J., Lassila, O.: The semantic web. Scientific American 284(5), 34–43 (2001)
2. Sycara, K.P., Paolucci, M., Ankolekar, A., Srinivasan, N.: Automated discovery, interaction and composition of semantic web services. J. Web Sem. 1(1), 27–46 (2003)
3. Smith, M.K., Welty, C., McGuinness, D.L.: Owl web ontology language guide. W3c recommendation, W3C (2004)
4. Baader, F., Nutt, W.: In: The Description Logic Handbook: Theory, Implementation, and Applications (2003)
5. Pistore, M., Marconi, A., Bertoli, P., Traverso, P.: Automated composition of web services by planning at the knowledge level. In: IJCAI, pp. 1252–1259 (2005)
6. Hoffmann, J., Bertoli, P., Helmert, M., Pistore, M.: Message-based web service composition, integrity constraints, and planning under uncertainty: A new connection. J. Artif. Intell. Res. (JAIR) 35, 49–117 (2009)
7. Marconi, A., Pistore, M., Traverso, P.: Implicit vs. explicit data-flow requirements in web service composition goals. In: Dan, A., Lamersdorf, W. (eds.) ICSOC 2006. LNCS, vol. 4294, pp. 459–464. Springer, Heidelberg (2006)
8. Papazoglou, M.P., Traverso, P., Dustdar, S., Leymann, F., Krämer, B.J.: 05462 service-oriented computing: A research roadmap. In: Service Oriented Computing (2005)
9. Kopecký, J., Vitvar, T., Bournez, C., Farrell, J.: Sawsdl: Semantic annotations for wsdl and xml schema. IEEE Internet Computing 11(6), 60–67 (2007)
10. Ankolenkar, A., Paolucci, M., Srinivasan, N., Sycara, K.: The owl-s coalition, owl-s 1.1. Technical report (2004)
11. Fensel, D., Kifer, M., de Bruijn, J., Domingue, J.: Web service modeling ontology submission, w3c submission (2005)
12. Ragone, A., Noia, T.D., Sciascio, E.D., Donini, F.M., Colucci, S., Colasuonno, F.: Fully automated web services discovery and composition through concept covering and concept abduction. Int. J. Web Service Res. 4(3), 85–112 (2007)
13. Ennals, R., Brewer, E.A., Garofalakis, M.N., Shadle, M., Gandhi, P.: Intel mash maker: join the web. SIGMOD Record 36(4), 27–33 (2007)
14. Phuoc, D.L., Polleres, A., Hauswirth, M., Tummarello, G., Morbidoni, C.: Rapid prototyping of semantic mash-ups through semantic web pipes. In: WWW, pp. 581–590 (2009)
15. Hull, D., Zolin, E., Bovykin, A., Horrocks, I., Sattler, U., Stevens, R.: Deciding semantic matching of stateless services. In: AAAI (2006)

16. Ambite, J.L., Kapoor, D.: Automatically composing data workflows with relational descriptions and shim services. In: Aberer, K., Choi, K.-S., Noy, N., Allemang, D., Lee, K.-I., Nixon, L.J.B., Golbeck, J., Mika, P., Maynard, D., Mizoguchi, R., Schreiber, G., Cudré-Mauroux, P. (eds.) ASWC 2007 and ISWC 2007. LNCS, vol. 4825, pp. 15–29. Springer, Heidelberg (2007)
17. Baader, F., Brandt, S., Lutz, C.: Pushing the el envelope. In: IJCAI, pp. 364–369 (2005)
18. Benatallah, B., Hacid, M., Leger, A., Rey, C., Toumani, F.: On automating web services discovery. VLDB Journal, 1–26 (December 2002)
19. Lécué, F., Léger, A.: A formal model for semantic web service composition. In: Cruz, I., Decker, S., Allemang, D., Preist, C., Schwabe, D., Mika, P., Uschold, M., Aroyo, L.M. (eds.) ISWC 2006. LNCS, vol. 4273, pp. 385–398. Springer, Heidelberg (2006)
20. Euzenat, J.: Semantic precision and recall for ontology alignment evaluation. In: IJCAI, pp. 348–353 (2007)
21. Paolucci, M., Kawamura, T., Payne, T., Sycara, K.: Semantic matching of web services capabilities. In: Horrocks, I., Hendler, J. (eds.) ISWC 2002. LNCS, vol. 2342, pp. 333–347. Springer, Heidelberg (2002)
22. Li, L., Horrocks, I.: A software framework for matchmaking based on semantic web technology. In: WWW, pp. 331–339 (2003)
23. Colucci, S., Noia, T.D., Sciascio, E.D., Donini, F.M., Mongiello, M.: Concept abduction and contraction in description logics. In: DL (2003)
24. Colucci, S., Noia, T.D., Sciascio, E.D., Donini, F.M., Mongiello, M.: A uniform tableaux-based method for concept abduction and contraction in description logics. In: ECAI, pp. 975–976 (2004)
25. Lécué, F., Delteil, A.: Making the difference in semantic web service composition. In: AAAI, pp. 1383–1388 (2007)
26. Noia, T.D., Sciascio, E.D., Donini, F.M.: Semantic matchmaking as non-monotonic reasoning: A description logic approach. J. Artif. Intell. Res. (JAIR) 29, 269–307 (2007)
27. Baader, F.: Terminological cycles in a description logic with existential restrictions. In: IJCAI, pp. 325–330 (2003)
28. Noia, T.D., Sciascio, E.D., Donini, F.M., Mongiello, M.: Abductive matchmaking using description logics. In: IJCAI, pp. 337–342 (2003)
29. Wu, D., Parsia, B., Sirin, E., Hendler, J.A., Nau, D.S.: Automating DAML-S web services composition using SHOP2. In: Fensel, D., Sycara, K., Mylopoulos, J. (eds.) ISWC 2003. LNCS, vol. 2870, pp. 195–210. Springer, Heidelberg (2003)
30. Oh, S.C., Kil, H., Lee, D., Kumara, S.R.T.: Wsben: A web services discovery and composition benchmark. In: ICWS, pp. 239–248 (2006)
31. Daniel, F., Casati, F., Benatallah, B.: Hosted universal composition: Models, languages and infrastructure in mashart. In: ER, pp. 428–443 (2009)
32. Krummenacher, R., Norton, B., Marte, A.: Towards linked open services and processes. In: Berre, A.J., Gómez-Pérez, A., Tutschku, K., Fensel, D. (eds.) FIS 2010. LNCS, vol. 6369, pp. 68–77. Springer, Heidelberg (2010)
33. McIlraith, S.A., Son, T.C.: Adapting golog for composition of semantic web services. In: KR, pp. 482–496 (2002)
34. Stuckenschmidt, H.: Partial matchmaking using approximate subsumption. In: AAAI, pp. 1459–1464 (2007)
35. McGuinness, D.L., Borgida, A.: Explaining subsumption in description logics. In: IJCAI (1), pp. 816–821 (1995)

The Combined Approach to OBDA: Taming Role Hierarchies Using Filters

Carsten Lutz[1], İnanç Seylan[1], David Toman[2], and Frank Wolter[3]

[1] Universität Bremen, Germany
{clu,seylan}@informatik.uni-bremen.de
[2] Cheriton School of CS, University of Waterloo, Canada
david@cs.uwaterloo.ca
[3] University of Liverpool, United Kingdom
wolter@liverpool.ac.uk

Abstract. The basic idea of the combined approach to query answering in the presence of ontologies is to materialize the consequences of the ontology in the data and then use a limited form of query rewriting to deal with infinite materializations. While this approach is efficient and scalable for ontologies that are formulated in the basic version of the description logic DL-Lite, it incurs an exponential blowup during query rewriting when DL-Lite is extended with the popular role hierarchies. In this paper, we show how to replace the query rewriting with a filtering technique. This is natural from an implementation perspective and allows us to handle role hierarchies without an exponential blowup. We also carry out an experimental evaluation that demonstrates the scalability of this approach.

1 Introduction

In recent years, ontology-based data access (OBDA) has emerged as a promising and challenging application of ontologies. The idea is to enrich data with a 'semantic layer' in the form of an ontology, used as an interface for querying and to derive additional answers. A central research problem in this area is to design query answering engines that can deal with sufficiently expressive ontology languages yet scale to very large data sets. The most popular ontology languages that have been considered for OBDA include the three OWL profiles OWL2 RL, OWL2 QL, and OWL2 EL, as well as various description logics and Datalog variants related to these profiles [2,3,5,14,17].

Currently, there are two major methodologies for answering queries in an OBDA setting: rewriting-based approaches (also called backward chaining) and materialization-based approaches (also called forward chaining). In the former, one compiles the ontology $\mathcal{T}$ and the query q into a new query $q_{\mathcal{T}}$ that contains the relevant knowledge from the ontology, i.e., the answers to q over $\mathcal{A}$ and $\mathcal{T}$ coincide with the answers to $q_{\mathcal{T}}$ over $\mathcal{A}$. One can thus store $\mathcal{A}$ in a relational database management system (RDBMS) and execute $q_{\mathcal{T}}$ over $\mathcal{A}$. In materialization approaches, the data $\mathcal{A}$ is completed with the relevant knowledge from

H. Alani et al. (Eds.): ISWC 2013, Part I, LNCS 8218, pp. 314–330, 2013.

the ontology $\mathcal{T}$, i.e., for any query q, the answers given to q over $\mathcal{A}$ and $\mathcal{T}$ coincide with the answers given to q over the completed data $\mathcal{A}_\mathcal{T} \supseteq \mathcal{A}$ without any ontology. Thus, one can store $\mathcal{A}_\mathcal{T}$ in a RDBMS and execute q over $\mathcal{A}_\mathcal{T}$.

A technical problem that arises in materialization approaches is that the completed data $\mathcal{A}_\mathcal{T}$ easily becomes infinite; in particular, this may happen when the ontology expresses cyclic dependencies and has existential quantifiers in the heads of its concept inclusions, which is allowed in most ontology languages including the ones mentioned above. To overcome this problem, an economic way of reusing individuals introduced for existential quantifiers has been proposed in [9,11] for the case where ontologies are formulated in description logics from the $\mathcal{EL}$ and DL-Lite families, which are the logical cores of the OWL2 EL and OWL2 QL ontology languages. While the resulting completed data sets are finite, they can give spurious answers to conjunctive queries (CQs) that involve a cycle. To recover soundness, it is thus necessary to include an additional step, resulting in the *combined approach* to query answering: the original query is rewritten in a way that eliminates spurious answers. In contrast to pure rewriting, the *auxiliary query rewriting* required in the combined approach turns out to be rather simple—an additional *selection condition* applied to the results of the original CQ over the completed data—and often of polynomial size. Indeed, experiments indicate that the combined approach admits very efficient query execution for expressive variants of $\mathcal{EL}$ and DL-Lite [9,11].

Unfortunately, there are certain combinations of logical operators that are important from an application perspective, but for which an exponential blowup of the query seems to be unavoidable both in the query rewriting approach and in the combined approach. In particular, this is the case for the combination of inverse roles and role hierarchies as found in DL-Lite$_\mathcal{R}$ [3], the extension of basic DL-Lite with role hierarchies that underpins OWL2 QL. It has been shown that, in the query rewriting approach, an exponential blowup of the query size is unavoidable when the ontology is formulated in DL-Lite$_\mathcal{R}$ [8]. For the combined approach, an auxiliary query rewriting strategy for DL-Lite$_\mathcal{R}$ ontologies and CQs is presented in [9], but it incurs an exponential blowup and it seems unlikely that the rewriting can be improved to a poly-sized one (although this question is yet to be resolved).

In this paper, we present a new variation on the combined approach that can handle CQs and DL-Lite$_\mathcal{R}$ ontologies and eliminates the need for auxiliary query rewriting altogether, thus also eliminating the need to deal with exponentially sized queries. Specifically, we replace auxiliary query rewriting with a *filtering component*: spurious answers are eliminated by a polynomial-time filtering procedure (called a *filter* in the rest of the paper) that is installed as a user-defined function in the underlying RDBMS. Our main contributions are as follows.

(1) We develop a polynomial time procedure for filtering out spurious answers to CQs for ontologies formulated in DL-Lite$_\mathcal{R}$. Interestingly, the existence of such a filtering procedure appears to be quite sensitive to how exactly the data is completed. Compared to the data completion for the original combined

approach [9], the filtering technique requires subtle modifications to the data completion in order to obtain a polytime filter.

(2) To analyze the performance of our approach and to compare it with the query rewriting approach, we modify the Lehigh University Benchmark (LUBM) [7] by introducing additional concepts into the ontology, modifying the data generator so that the produced data is incomplete, and replacing the original, very simple queries by more challenging ones.

(3) We have implemented our approach in a system called COMBO and carry out an experimental evaluation based on the modified LUBM benchmark, both to evaluate the feasibility of our approach and to compare it with the query rewriting approach. Our experiments show that the combined approach is significantly more robust than the rewriting approach when the number of (sub)classes in the ontology or the size of the data increases.

Some technical proofs and details of our experimental evaluation are presented in the appendix of the full version of this paper, available at `http://www.informatik.uni-bremen.de/~clu/combined/`. This paper is an extended version of the workshop paper [10]. In particular, the experimental evaluation carried out in this paper is much more comprehensive than the one in [10].

2 Preliminaries

We introduce DL-Lite$_\mathcal{R}$-TBoxes, ABoxes, and conjunctive queries. Let $\mathsf{N_I}$, $\mathsf{N_C}$, and $\mathsf{N_R}$ be countably infinite sets of *individual names*, *concept names* and *role names*. *Roles* R, *simple concepts* C, and *concepts* D are built according to the following syntax rules, where P ranges over $\mathsf{N_R}$ and A over $\mathsf{N_C}$:

$$R ::= P \mid P^-, \qquad C ::= A \mid \exists R, \qquad D ::= C \mid \neg C \mid \exists R.A.$$

As usual, we use $\mathsf{N_R^-}$ to denote the set of all roles and identify $(P^-)^-$ with P. In DL-Lite$_\mathcal{R}$, a *TBox* is a finite set $\mathcal{T}$ of *concept inclusions (CIs)* $C \sqsubseteq D$ with C a simple concept and D a concept, and *role inclusions (RIs)* $R_1 \sqsubseteq R_2$ with R_1, R_2 roles.

An *ABox* is a finite set of *concept assertions* $A(a)$ and *role assertions* $P(a, b)$, where $A \in \mathsf{N_C}$, $P \in \mathsf{N_R}$ and $a, b \in \mathsf{N_I}$. We denote by $\mathsf{Ind}(\mathcal{A})$ the set of individual names used in $\mathcal{A}$, and write $P^-(a, b) \in \mathcal{A}$ instead of $P(b, a) \in \mathcal{A}$ if convenient. A *knowledge base* (*KB*) is a pair $(\mathcal{T}, \mathcal{A})$ with $\mathcal{T}$ a TBox and $\mathcal{A}$ an ABox.

The semantics of TBoxes and ABoxes is defined in the standard way based on interpretations $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$, where $\Delta^\mathcal{I}$ is a non-empty *domain* and $\cdot^\mathcal{I}$ an *interpretation function* that maps each $A \in \mathsf{N_C}$ to a subset $A^\mathcal{I} \subseteq \Delta^\mathcal{I}$, each $P \in \mathsf{N_R}$ to a relation $P^\mathcal{I} \subseteq \Delta^\mathcal{I} \times \Delta^\mathcal{I}$, and each $a \in \mathsf{N_I}$ to an element $a^\mathcal{I} \in \Delta^\mathcal{I}$; for details consult [1,3]. An interpretation is a *model* of a TBox $\mathcal{T}$ if it satisfies all inclusions in $\mathcal{T}$; models of ABoxes and knowledge bases are defined analogously. A knowledge base is *consistent* if it has a model. For a CI or RI α, we write $\mathcal{T} \models \alpha$ when α is a consequence of $\mathcal{T}$ (satisfied in all models of $\mathcal{T}$). Instead of

$\mathcal{T} \models R \sqsubseteq S$, we usually write $R \sqsubseteq^*_{\mathcal{T}} S$ to clearly distinguish consequences of this form (which are RIs) from consequences of the form $\mathcal{T} \models \exists R \sqsubseteq \exists S$ (which are CIs). Note that, in DL-Lite$_{\mathcal{R}}$, deciding consistency and logical consequence amounts to computing a form of transitive closure [3].

Let $\mathsf{N_V}$ be a countably infinite set of *variables*. Taken together, the sets $\mathsf{N_V}$ and $\mathsf{N_I}$ form the set $\mathsf{N_T}$ of *terms*. A *conjunctive query (CQ)* takes the form $q = \exists \boldsymbol{y}\, \psi(\boldsymbol{y}, \boldsymbol{x})$, where ψ is a conjunction of *concept atoms* $A(t)$ and *role atoms* $P(t, t')$ where $t, t' \in \mathsf{N_T}$. As in the case of ABox assertions, we do not distinguish between $P^-(t, t')$ and $P(t', t)$. The free variables $\boldsymbol{x}$ of φ are called the *answer variables*; we say that q is k*-ary* if $\boldsymbol{x}$ comprises k variables. If $k = 0$, then q is a *Boolean* query. A *union of conjunctive queries (UCQ)* is a disjunction of CQs. We denote by $\mathsf{term}(q)$ the set of terms in q.

Let $q = \exists \boldsymbol{y}\, \psi(\boldsymbol{y}, \boldsymbol{x})$ be a k-ary CQ with $\boldsymbol{x} = x_1, \ldots, x_k$, and $\mathcal{I}$ an interpretation. A mapping $\pi\colon \mathsf{term}(q) \to \Delta^{\mathcal{I}}$ with $\pi(a) = a^{\mathcal{I}}$ for all $a \in \mathsf{term}(q) \cap \mathsf{N_I}$ is a *match* for q in $\mathcal{I}$ if $\mathcal{I}$ satisfies ψ under the variable assignment that maps each $t \in \mathsf{term}(q)$ to $\pi(t)$; in this case, we write $\mathcal{I} \models^{\pi} q$. For a k-tuple of individual names $\boldsymbol{a} = a_1, \ldots, a_k$, a match π for q in $\mathcal{I}$ is an $\boldsymbol{a}$*-match* if $\pi(x_i) = a_i^{\mathcal{I}}$ for $i \leq k$. We say that $\boldsymbol{a}$ is an *answer* to q in an interpretation $\mathcal{I}$ if there is an $\boldsymbol{a}$-match for q in $\mathcal{I}$ and use $\mathsf{ans}(q, \mathcal{I})$ to denote the set of all answers to q in $\mathcal{I}$. Finally, $\boldsymbol{a}$ is a *certain answer* to q over a KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ if $\boldsymbol{a} \subseteq \mathsf{Ind}(\mathcal{A})$ and $\mathcal{I} \models q[\boldsymbol{a}]$ for all models $\mathcal{I}$ of $\mathcal{K}$. The set of all certain answers to q over $\mathcal{K}$ is denoted by $\mathsf{cert}(q, \mathcal{K})$. The query answering problem considered in this paper is: given a DL-Lite$_{\mathcal{R}}$ knowledge base $\mathcal{K}$ and a CQ q, compute $\mathsf{cert}(q, \mathcal{K})$.

To simplify notation, throughout the paper we adopt the *unique name assumption* (UNA), i.e., require that $a^{\mathcal{I}} \neq b^{\mathcal{I}}$ for distinct $a, b \in \mathsf{N_I}$. This assumption has no impact on the query answering problem.

3 ABox Completion

As explained in the introduction, the central idea of the combined approach is to materialize consequences of the TBox in the ABox as a preprocessing step, and then to execute queries over the completed data stored in an RDBMS as a plain table. We illustrate this using two examples from the university domain, similar in spirit to the LUBM ontology used in the experimental evaluation.

Example 1. For any ABox $\mathcal{A}$, the concept inclusions

$$\mathsf{Student} \sqsubseteq \mathsf{Person} \tag{1}$$

$$\mathsf{Student} \sqsubseteq \exists \mathsf{takesCourse} \tag{2}$$

lead to the following additions: (1) for every assertion $\mathsf{Student}(a) \in \mathcal{A}$, add (1) $\mathsf{Person}(a)$ and (2) $\mathsf{takesCourse}(a, b)$ for some fresh individual b (unless such assertions are already present). After this completion, a CQ such as

$$q_1(x) = \exists y\, \mathsf{Person}(x) \wedge \mathsf{takesCourse}(x, y)$$

correctly returns each a with $\mathsf{Student}(a) \in \mathcal{A}$ as a certain answer.

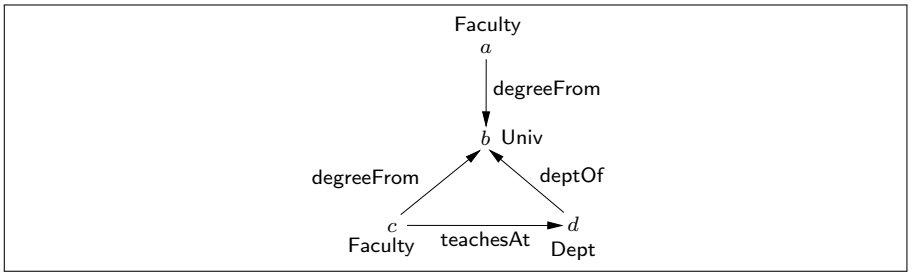


Fig. 1. Completed ABox for Example 3

The following example shows that naive completion can result in infinite ABoxes.

Example 2. Completed naively, the ABox $\{\mathsf{Faculty}(a)\}$ *and LUBM inclusions*

$$\mathsf{Faculty} \sqsubseteq \exists \mathsf{degreeFrom} \qquad \exists \mathsf{degreeFrom}^- \sqsubseteq \mathsf{Univ} \tag{3}$$

$$\mathsf{Univ} \sqsubseteq \exists \mathsf{deptOf}^- \qquad \exists \mathsf{deptOf} \sqsubseteq \mathsf{Dept} \tag{4}$$

$$\mathsf{Dept} \sqsubseteq \exists \mathsf{teachesAt}^- \qquad \exists \mathsf{teachesAt} \sqsubseteq \mathsf{Faculty} \tag{5}$$

result in an infinite role chain that indefinitely repeats the roles $\mathsf{degreeFrom}$, deptOf^-, *and* $\mathsf{teachesAt}^-$.

The problem can be overcome by reusing fresh individuals in an economic way.

Example 3. Consider again the TBox (3)-(5). By reusing individuals, the ABox $\{\mathsf{Faculty}(a)\}$ *can be completed as shown in Figure 1, replacing the infinite role chain with a cycle. Individual reuse compromises soundness of query answering as some queries now have spurious answers; for example, the CQ*

$$q_2(x) = \exists y, z\ \mathsf{Faculty}(x) \wedge \mathsf{degreeFrom}(x, y) \wedge \mathsf{Univ}(y) \wedge \mathsf{deptOf}(z, y) \wedge \mathsf{Dept}(z) \wedge \mathsf{teachesAt}(x, z)$$

returns c as an answer when executed over the completed ABox shown in Figure 1. This answer is spurious for two reasons: first, the cycle in Figure 1 is present only due to individual reuse and thus should be disregarded for answering queries; and second, the freshly introduced individuals b, c, d *are 'labeled nulls' and thus can never be returned as answers.*

To recover soundness, it is necessary to eliminate the spurious answers. In the original combined approach, this was achieved by query rewriting [9,11]. In this paper, the spurious answers are eliminated by a filtering procedure that is installed as a user-defined function in the RDBMS. In the remainder of this section, we introduce ABox completion in full detail. In the subsequent section, we describe the filtering procedure.

From a conceptual perspective, the ABox completion step can be viewed as replacing the original ABox with the *canonical model* $\mathcal{I}_\mathcal{K}$ of the knowledge base $\mathcal{K}$ [9]. To define $\mathcal{I}_\mathcal{K}$, we need a few preliminaries. From now on, we will

generally disallow concepts of the form $\exists R.C$. This can be done without loss of generality since each CI $D \sqsubseteq \exists R.C$ can be replaced with $D \sqsubseteq \exists R_C$, $R_C \sqsubseteq R$, and $\exists R_C^- \sqsubseteq C$, where R_C is a fresh role name.

Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be a DL-Lite$_\mathcal{R}$ KB. We use $\mathsf{rol}(\mathcal{T})$ to denote the set of all role names in $\mathcal{T}$ plus their inverses. The canonical model comprises at most two fresh individuals for every role in $\mathsf{rol}(\mathcal{T})$. However, we only want to introduce the fresh individuals for a given role when necessary. Formally, we call a role $R \in \mathsf{rol}(\mathcal{T})$ *generating in* $\mathcal{K}$ if there exist an $a \in \mathsf{Ind}(\mathcal{A})$ and $R_0, \ldots, R_n \in \mathsf{rol}(\mathcal{T})$ such that $R_n = R$ and the following conditions hold:

(agen) $\mathcal{K} \models \exists R_0(a)$ and $R_0(a, b) \notin \mathcal{A}$ for all $b \in \mathsf{Ind}(\mathcal{A})$ (written $a \rightsquigarrow \exists R_0$),
(rgen) for $i < n$, $\mathcal{T} \models \exists R_i^- \sqsubseteq \exists R_{i+1}$ and $R_i^- \neq R_{i+1}$ (written $\exists R_i^- \rightsquigarrow \exists R_{i+1}$).

To facilitate the implementation of efficient filters, we refine the definition of canonical models as given in [9]: in some cases, we introduce two fresh individuals for a given role instead of only a single one. This helps to avoid choices in the elimination of spurious answers (see Example 8), which are related to particular role configurations in the TBox called a loop: a set $\{R, S\} \subseteq \mathsf{rol}(\mathcal{T})$ (where potentially $R = S$) is a *loop in* $\mathcal{T}$ if $R \neq S^-$, $\mathcal{T} \models \exists R^- \sqsubseteq \exists S$, $\mathcal{T} \models \exists S^- \sqsubseteq \exists R$, and there is some $T \in \mathsf{rol}(\mathcal{T})$ such that $S^- \sqsubseteq_\mathcal{T}^* T$ and $R \sqsubseteq_\mathcal{T}^* T$. Let $\mathfrak{L}_\mathcal{T}$ denote the set of all roles that occur in a loop in $\mathcal{T}$. The canonical model $\mathcal{I}_\mathcal{K}$ is then based on the domain

$$\begin{aligned}\Delta^{\mathcal{I}_\mathcal{K}} = \mathsf{Ind}(\mathcal{A}) \cup \{c_{R,0} \mid R \in \mathsf{rol}(\mathcal{T}) \setminus \mathfrak{L}_\mathcal{T} \text{ is generating in } \mathcal{K}\} \\ \cup \{c_{R,0}, c_{R,1} \mid R \in \mathfrak{L}_\mathcal{T} \text{ is generating in } \mathcal{K}\}.\end{aligned}$$

To define the extension of roles in $\mathcal{I}_\mathcal{K}$, we need some additional preparation. Let "$\prec$" be an arbitrary, but fixed total ordering on $\mathsf{rol}(\mathcal{T})$. For all $d, d' \in \Delta^{\mathcal{I}_\mathcal{K}}$ and each role R, we write $d \rightsquigarrow_R d'$ whenever there is an S such that $S \sqsubseteq_\mathcal{T}^* R$ and one of the following cases applies:

- $d = a \in \mathsf{Ind}(\mathcal{A})$, $a \rightsquigarrow \exists S$, and $d' = c_{S,0}$;
- $d = c_{T,i}$, $\exists T^- \rightsquigarrow \exists S$, $d' = c_{S,j}$, and one of the following holds
  - $i = j$ and $\{S, T\}$ is not a loop in $\mathcal{T}$;
  - $i = j$, $\{S, T\}$ is a loop in $\mathcal{T}$, and $S \prec T$;
  - $i = \overline{j}$, $\{S, T\}$ is a loop in $\mathcal{T}$, and $T = S$ or $T \prec S$ (for $\overline{0} = 1$ and $\overline{1} = 0$).

The *canonical model* $\mathcal{I}_\mathcal{K}$ *for* $\mathcal{K}$ is now defined as follows, based on the domain $\Delta^{\mathcal{I}_\mathcal{K}}$ introduced above:

$$\begin{aligned}A^{\mathcal{I}_\mathcal{K}} &= \{a \in \mathsf{Ind}(\mathcal{A}) \mid \mathcal{K} \models A(a)\} \cup \{c_{R,i} \in \Delta^{\mathcal{I}_\mathcal{K}} \mid \mathcal{T} \models \exists R^- \sqsubseteq A\}, \\ R^{\mathcal{I}_\mathcal{K}} &= \{(a, b) \in \mathsf{Ind}(\mathcal{A}) \times \mathsf{Ind}(\mathcal{A}) \mid \exists S : S(a, b) \in \mathcal{A} \text{ and } S \sqsubseteq_\mathcal{T}^* R\} \cup \\ &\quad \{(d, d') \in \Delta^{\mathcal{I}_\mathcal{K}} \mid d \rightsquigarrow_R d' \text{ or } d' \rightsquigarrow_{R^-} d\}, \\ a^{\mathcal{I}_\mathcal{K}} &= a.\end{aligned}$$

Note that the slightly more straightforward version of canonical models defined in [9] can be obtained from our canonical models by identifying all elements $c_{R,0}$ and $c_{R,1}$.

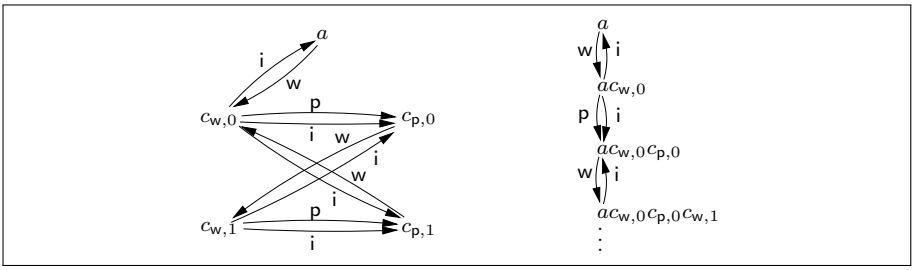


Fig. 2. Canonical model $\mathcal{I}_\mathcal{K}$ and unraveled canonical model $\mathcal{U}_\mathcal{K}$ for Example 5

The ABox completion consists of replacing the ABox $\mathcal{A}$ originally stored in the RDBMS with its canonical model $\mathcal{I}_\mathcal{K}$. This can be achieved by executing a set of FO/SQL-queries whose size is polynomial in the size of $\mathcal{T}$ [9].

It can be shown that $\mathcal{I}_\mathcal{K}$ is a model of $\mathcal{K}$ whenever $\mathcal{K}$ is consistent. Note that one can find a Boolean CQ $q_\mathcal{T}$ of size polynomial in the size of $\mathcal{T}$ such that for any ABox $\mathcal{A}$ stored in the RDBMS, $q_\mathcal{T}$ gives a positive answer iff $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ is consistent [9]. We can thus safely assume that the knowledge base has been tested for consistency before query answering.

Example 4. Reconsider Examples 2 and 3. The canonical model for the ABox $\{\mathsf{Faculty}(a)\}$ *and TBox (3)-(5) is the structure displayed in Figure 1. Following our construction above, the fresh individuals* b, c, d *are named* $c_{\mathsf{degreeFrom},0}$, $c_{\mathsf{teachesAt}^-,0}$, *and* $c_{\mathsf{deptOf}^-,0}$. *Note that the TBox (3)-(5) does not give rise to any loops, and thus all* $c_{R,i}$ *have index* $i = 0$.

Example 5. The following TBox gives rise to the loop $\{\mathsf{worksFor}, \mathsf{paysSalaryOf}\}$*:*

$$\mathsf{Employee} \sqsubseteq \exists \mathsf{worksFor} \qquad \exists \mathsf{worksFor}^- \sqsubseteq \mathsf{Employer} \tag{6}$$

$$\mathsf{Employer} \sqsubseteq \exists \mathsf{paysSalaryOf} \qquad \exists \mathsf{paysSalaryOf}^- \sqsubseteq \mathsf{Employee} \tag{7}$$

$$\mathsf{worksFor}^- \sqsubseteq \mathsf{isAffiliatedWith} \qquad \mathsf{paysSalaryOf} \sqsubseteq \mathsf{isAffiliatedWith}. \tag{8}$$

A part of the canonical model for the ABox $\{\mathsf{Employee}(a)\}$ *and the TBox (6)-(8) with* $\mathsf{paysSalaryOf} \prec \mathsf{worksFor}$ *is shown on the left-hand side of Figure 2, where concept names are omitted and role names are abbreviated by their first letter.*

To characterize the spurious answers that have to be filtered out, it is useful to introduce an unraveled (infinite) version of canonical models. Let $\mathcal{K}$ be a knowledge base. A *path* is a finite sequence $ad_1 \cdots d_n$, $n \geq 0$, such that $a \in \mathsf{Ind}(\mathcal{A})$, $d_1, \ldots, d_n \in \Delta^{\mathcal{I}_\mathcal{K}} \setminus \mathsf{Ind}(\mathcal{A})$, $a \rightsquigarrow_R d_1$ for some $R \in \mathsf{N}_\mathsf{R}^-$, and $d_i \rightsquigarrow_R d_{i+1}$ for some $R \in \mathsf{N}_\mathsf{R}^-$, $1 \leq i < n$. We denote by $\mathsf{tail}(\sigma)$ the last element of the path σ. The unraveled canonical model $\mathcal{U}_\mathcal{K}$ is then defined by taking:

$$\begin{aligned} \Delta^{\mathcal{U}_\mathcal{K}} &\text{ is the set of all paths in } \mathcal{I}_\mathcal{K}, \\ a^{\mathcal{U}_\mathcal{K}} &= a, \text{ for all } a \in \mathsf{Ind}(\mathcal{A}), \\ A^{\mathcal{U}_\mathcal{K}} &= \{\sigma \in \Delta^{\mathcal{U}_\mathcal{K}} \mid \mathsf{tail}(\sigma) \in A^{\mathcal{I}_\mathcal{K}}\}, \end{aligned}$$

$$R^{\mathcal{U}_\mathcal{K}} = \{(a,b) \in \mathsf{Ind}(\mathcal{A}) \times \mathsf{Ind}(\mathcal{A}) \mid \exists S : S(a,b) \in \mathcal{A} \text{ and } S \sqsubseteq^*_\mathcal{T} R\} \cup$$
$$\{(\sigma, \sigma d) \mid \sigma d \in \Delta^{\mathcal{U}_\mathcal{K}} \text{ and } \mathsf{tail}(\sigma) \rightsquigarrow_R d\} \cup$$
$$\{(\sigma d, \sigma) \mid \sigma d \in \Delta^{\mathcal{U}_\mathcal{K}} \text{ and } \mathsf{tail}(\sigma) \rightsquigarrow_{R^-} d\}.$$

As an example, the canonical model $\mathcal{U}_\mathcal{K}$ for the KB from Example 5 is shown on the right-hand side of Figure 2. The following result shows that, as one would expect, $\mathcal{U}_\mathcal{K}$ does not suffer from spurious answers.

Theorem 1. *For every consistent DL-Lite$_\mathcal{R}$-KB $\mathcal{K}$ and every CQ q, we have* $\mathsf{cert}(q,\mathcal{K}) = \mathsf{ans}(q,\mathcal{U}_\mathcal{K})$.

The proof of Theorem 1 is standard and omitted, see [9] for a similar proof.

4 Filtering

To remove spurious answers, we install a filtering procedure as a user-defined function in the RDBMS. In this approach, calls to the filtering procedure are delegated to the RDBMS in hopes that the query optimizer will eliminate spurious answers as early as possible in the execution plan. The procedure takes as input a match of the query in the canonical model $\mathcal{I}_\mathcal{K}$ stored in the RDBMS and returns "false" if this match is spurious and "true" otherwise. We assume that the filtering procedure has access to the query and the TBox, but not to the data. To define its behavior more precisely, we formally define spurious matches based on unraveled canonical models $\mathcal{U}_\mathcal{K}$ and Theorem 1.

Let $\mathcal{K}$ be a KB and $q(\boldsymbol{x})$ a CQ. A match π of q in $\mathcal{I}_\mathcal{K}$ *is reproduced by* a match τ of q in $\mathcal{U}_\mathcal{K}$ if for all $t \in \mathsf{term}(q)$, we have $\pi(t) = \mathsf{tail}(\tau(t))$. We say that π is *spurious* if it is not reproduced by any match τ of q in $\mathcal{U}_\mathcal{K}$. The following lemma, which is an immediate consequence of Theorem 1, shows that $\mathcal{I}_\mathcal{K}$ can be used for query answering when spurious matches are filtered out.

Lemma 1. $\boldsymbol{a} \in \mathsf{cert}(q,\mathcal{K})$ *iff there is a non-spurious* $\boldsymbol{a}$*-match* π *of* q *in* $\mathcal{I}_\mathcal{K}$.

We want to show that it can be decided in time polynomial in the size of q and $\mathcal{T}$ (and without accessing $\mathcal{A}$ at all) whether a given match in $\mathcal{I}_\mathcal{K}$ is spurious. Clearly, it is enough to test for each maximally connected component of q whether the match is spurious on that component. We thus assume that q is connected.

We need a few preliminaries. An *anonymous path* is a path without the leading individual name, i.e., it is a finite sequence $d_1 \cdots d_n$, $n \geq 1$, such that $d_1, \ldots, d_n \in \Delta^{\mathcal{I}_\mathcal{K}} \setminus \mathsf{Ind}(\mathcal{A})$ and $d_i \rightsquigarrow_R d_{i+1}$ for some $R \in \mathsf{N}_\mathsf{R}^-$, $1 \leq i < n$. We use Paths to denote the set of all paths, both anonymous and non-anonymous. A *root configuration for q given π* is a set $\rho \subseteq \mathsf{term}(q)$ such that one of the following conditions is true:

- ρ is the set of those $t \in \mathsf{term}(q)$ such that $\pi(t) \in \mathsf{N}_\mathsf{I}$ and this set is non-empty;
- the above set is empty and ρ contains exactly one term (actually a variable).

The filtering procedure immediately returns "false" if some answer variable is mapped to an element of $\Delta^{\mathcal{I}_\mathcal{K}}$ that is not from $\mathsf{Ind}(\mathcal{A})$ (based on the name of

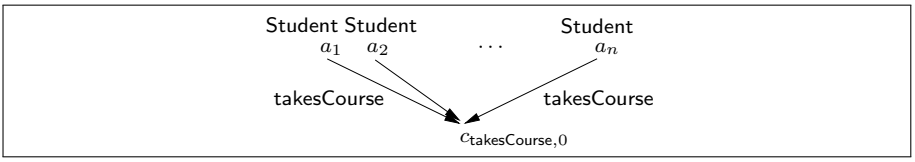


Fig. 3. Canonical model $\mathcal{I}_\mathcal{K}$ for Example 6

the element, i.e., whether it is of the form $c_{R,i}$). Then the procedure iterates through all root configurations ρ. For each ρ, it constructs a sequence $S^0_\rho, S^1_\rho, \ldots$ of relations $S^i_\rho \subseteq \mathsf{term}(q) \times \mathsf{Paths}$ as follows:

- S^0_ρ contains all pairs $(t, \pi(t))$ with $t \in \rho$;
- S^{i+1}_ρ is S^i_ρ extended with the following pairs:
  (a) $(t, \sigma\pi(t))$ for all $R(s,t) \in q$ with $(s,\sigma) \in S^i_\rho$ and $\pi(s) \rightsquigarrow_R \pi(t)$;
  (b) $(t, \sigma\pi(t))$ for all $R(s,t) \in q$ with $(s, \sigma\pi(t)\pi(s)) \in S^i_\rho$ and $\pi(t) \rightsquigarrow_{R^-} \pi(s)$.

The computation stops as soon as the sequence stabilizes or S^i_ρ becomes non-functional which happens after at most $|\mathsf{term}(q)|$ iterations. The procedure returns "true" if the final S^i_ρ is a function with domain $\mathsf{term}(q)$ for some root configuration ρ, and "false" otherwise.

Example 6. Consider the TBox (1)-(2) from Example 1, the query

$$q_3(x,y) = \exists z\, \mathsf{Student}(x) \wedge \mathsf{Student}(y) \wedge \mathsf{takesCourse}(x,z) \wedge \mathsf{takesCourse}(y,z), \quad (9)$$

and the ABox

$$\{\mathsf{Student}(a_1), \ldots, \mathsf{Student}(a_n)\}. \quad (10)$$

The canonical model $\mathcal{I}_\mathcal{K}$ is shown in Figure 3. Suppose the filter gets as input the match $\pi = \{x \mapsto a_1, y \mapsto a_2, z \mapsto c_{\mathsf{takesCourse},0}\}$. There is only one possible root configuration for π, which is $\rho = \{x, y\}$. The procedure computes

$$S_\rho = \{(x, a_1), (y, a_2), (z, a_1 c_{\mathsf{takesCourse},0}), (z, a_2 c_{\mathsf{takesCourse},0})\}$$

which is not a function; thus, the match is spurious and "false" is returned.

Example 7. Consider the ABox $\{\mathsf{Faculty}(a)\}$, TBox (3)-(5), and query q_2 from Example 3. To make things a bit more interesting, assume that x is a quantified variable in q_2 rather than an answer variable. Recall that the canonical model $\mathcal{I}_\mathcal{K}$ is shown in Figure 1, modulo the names of fresh individuals. Given the match $\pi = \{x \mapsto c, y \mapsto b, z \mapsto d\}$ and considering the root configuration $\rho = \{x\}$, the procedure computes

$$S_\rho = \{(x, c), (y, cb), (z, cbd), (x, cbdc)\}$$

and stops because of non-functionality. For the other root configurations $\rho = \{y\}$ and $\rho = \{z\}$, the procedure fails in a similar way and thus returns "false".

Similar to the "tree witnesses" from [9], the filtering procedure follows a simple idea for reproducing the input match π in $\mathcal{I}_\mathcal{K}$ as a match τ in $\mathcal{U}_\mathcal{K}$: when we have already decided that $\tau(x) = \sigma \notin \mathsf{Ind}(\mathcal{A})$ and $R(x, y) \in q$, then there is a *uniquely determined* individual σ' to which y can be matched. This follows from requiring $\pi(y) = \mathsf{tail}(\tau(y))$ and the following property of $\mathcal{U}_\mathcal{K}$:

$$\text{if } (\sigma, \sigma') \in R^{\mathcal{U}_\mathcal{K}} \text{ and } (\sigma, \sigma'') \in R^{\mathcal{U}_\mathcal{K}} \text{ with } \sigma' \neq \sigma'', \text{ then } \mathsf{tail}(\sigma') \neq \mathsf{tail}(\sigma'').$$

In fact, it is this determinism of matches that is made explicit by Conditions (a) and (b) of the filtering procedure. Note that, without introducing two individual names $c_{R,0}$ and $c_{R,1}$ whenever R is involved in a loop, the above crucial property of $\mathcal{U}_\mathcal{K}$ fails. In fact, we do not know whether polytime filtering is possible based on the variation of the canonical model where all individuals $c_{R,0}$ and $c_{R,1}$ are identified. The problem is illustrated by the following example.

Example 8. Consider the ABox $\{\mathsf{Employee}(a)\}$ *and TBox (6)-(8) from Example 5 and the CQ*

$$q_4(x) = \exists y, z, u\ \mathsf{w}(x, y) \land \mathsf{p}(y, z) \land \mathsf{i}(u, z).$$

Let $\pi = \{x \mapsto a, y \mapsto c_{\mathsf{w},0}, z \mapsto c_{\mathsf{p},0}, u \mapsto c_{\mathsf{w},1}\}$. *The only root configuration is* $\rho = \{x\}$. *During the first two iterations, the filtering procedure produces*

$$S^2_\rho = \{(x, a), (y, ac_{\mathsf{w},0}), (z, ac_{\mathsf{w},0}c_{\mathsf{p},0})\}.$$

S^2_ρ *says that* z *has to be mapped to* $ac_{\mathsf{w},0}c_{\mathsf{p},0}$. *Due to the atom* $\mathsf{i}(u, z) \in q_4$ *and the two* i*-edges incoming to* $ac_{\mathsf{w},0}c_{\mathsf{p},0}$ *in* $\mathcal{U}_\mathcal{K}$, *the possible targets for* u *are* $ac_{\mathsf{w},0}$ *and* $ac_{w,0}c_{p,0}c_{\mathsf{w},1}$. *However, to produce a match in* $\mathcal{U}_\mathcal{K}$ *that is compatible with* π, *we can only choose a target that ends with* $\pi(u) = c_{\mathsf{w},1}$ *and obtain*

$$S^3_\rho = \{(x, a), (y, ac_{\mathsf{w},0}), (z, ac_{\mathsf{w},0}c_{\mathsf{p},0}), (u, ac_{\mathsf{w},0}c_{\mathsf{p},0}c_{\mathsf{w},1})\}$$

which is functional, showing that the match π *is not spurious. In a canonical model* $\mathcal{I}_\mathcal{K}$ *where* $c_{\mathsf{w},0}$ *and* $c_{\mathsf{w},1}$ *are identified, there are indeed two choices for mapping of* u. *This makes it non-obvious how to find a polytime filtering procedure in this case, if one exists at all.*

We now analyze the runtime and correctness of the filtering procedure. First note that, in Conditions (a) and (b), the filtering procedure has to check whether $\pi(s) \rightsquigarrow_R \pi(t)$ and $\pi(t) \rightsquigarrow_{R^-} \pi(s)$, respectively. As required, both conditions can be tested without access to the ABox $\mathcal{A}$. For example, in Condition (a) we have:

- if $\pi(t) \in \mathsf{Ind}(\mathcal{A})$, then $\pi(s) \rightsquigarrow_R \pi(t)$ does not hold and checking whether $\pi(t) \in \mathsf{Ind}(\mathcal{A})$ requires only to check whether or not $\pi(t)$ is of the form $c_{R,i}$;
- if $\pi(s) \in \mathsf{Ind}(\mathcal{A})$ and $\pi(t) \notin \mathsf{Ind}(\mathcal{A})$, then $\pi(s) \rightsquigarrow_R \pi(t)$ holds by the construction of $\mathcal{I}_\mathcal{K}$ since π is a match of q in $\mathcal{I}_\mathcal{K}$, and;
- if $\pi(s) \notin \mathsf{Ind}(\mathcal{A})$ and $\pi(t) \notin \mathsf{Ind}(\mathcal{A})$, then $\pi(s) \rightsquigarrow_R \pi(t)$ can be checked by using only π and $\mathcal{T}$ based on the definition of "$\rightsquigarrow_R$".

It is not hard to see that the algorithm runs in polynomial time. The runtime is quadratic in the size of q because we first have to iterate over all root configurations ρ and then need to compute S_ρ, essentially a breadth-first search of (the graph of) q. We conjecture that iterating over all root configurations is avoidable at the cost of a less transparent filtering procedure, improving the runtime to linear in the size of q. The runtime also depends on $\mathcal{T}$ as checking the applicability of Conditions (a) and (b) involves testing consequences of the forms $\mathcal{T} \models \exists R \sqsubseteq \exists S$ and $S \sqsubseteq^*_{\mathcal{T}} R$. Since it is efficient to pre-compute all these consequences in practical cases, this amounts to a simple lookup.

The following lemma asserts correctness of the filtering procedure. It is proved in the appendix of the full version.

Lemma 2. *Given a match π of q in $\mathcal{I}_\mathcal{K}$, the filtering procedure returns "true" iff π is not spurious.*

5 Implementation and Experiments

We have implemented our approach in the Combo system, a collection of tools that support the user in setting up the tables of a relational database system to store ABoxes and their completion, implements the actual data completion via querying, and allows to compile an ontology into a filter that takes the form of a user defined function. The preferred relational database system for use with Combo is IBM DB2.

We use this combination to carry out an experimental evaluation of our approach, and to compare it to the query rewriting approach. The experiments are based on a modified version of the ontology from the Lehigh University Benchmark (LUBM) [7] and on ABoxes produced by a modified version of the LUBM data generator. We use six queries that were hand-crafted specifically for our experiments. The mentioned modifications aim at making the LUBM suite more realistic for OBDA evaluation. We believe that this setup might be interesting also for future experiments and provide it online at `http://www.informatik.uni-bremen.de/~clu/combined/`.

Regarding the query rewriting approach, we use Rapid (v0.3) [4] and Presto (version March 25th 2013) [15] as typical examples of state-of-the-art rewriting tools. Both Rapid and Presto are able to generate rewritings into UCQs and into non-recursive Datalog (DLog), and they use various optimizations to generate as small rewritings as possible.

5.1 Ontology, Data, and Queries

The LUBM ontology comprises 42 concept names and 25 role names and is formulated in the description logic $\mathcal{ELI}$ extended with transitive roles, role hierarchies, and datatypes. The TBox contains concept inclusions of the form $A \sqsubseteq C$, concept definitions $A \equiv C$ as abbreviations for $A \sqsubseteq C$, $C \sqsubseteq A$, and domain and range restrictions of the form $\exists R \sqsubseteq A$ and $\exists R^- \sqsubseteq A$. We converted this ontology to DL-Lite$_\mathcal{R}$ by dropping all datatypes, treating the only transitive role

subOrganizationOf as a standard role, replacing concept equations $A \equiv C$ with $A \sqsubseteq C$, and breaking up conjunctions $A \sqsubseteq C_1 \sqcap C_2$ into $A \sqsubseteq C_1, A \sqsubseteq C_2$.

While the resulting TBox is formulated in DL-Lite$_\mathcal{R}$ as required, it is only moderately interesting for evaluating query answering techniques: first, there is a lack of existential restrictions $\exists R$ and $\exists R.C$ on the right-hand side of concept inclusions, which leads to extremely few fresh *anonymous* individuals being generated during the ABox completion, and consequently to very few role edges between those individuals (from now on, we call this part of the canonical model $\mathcal{I}_\mathcal{K}$ the *anonymous part*); second, the overall size of the TBox is too small to be representative for real-world ontologies. To attenuate these deficiencies while still being able to use the LUBM data generator, we extended the DL-Lite$_\mathcal{R}$-version of LUBM in two directions:

(1) We added 26 carefully chosen concept inclusions, many of which have existential restrictions on the right-hand side, to generate a more interesting anonymous part of canonical models. A complete list of these CIs can be found in the appendix of the full version of this paper.

(2) With reasonable effort, it does not seem possible to significantly increase the size of LUBM (to hundreds or thousands of concepts) while retaining a careful modeling. One particularly unrealistic aspect of LUBM and a striking difference to many real-world ontologies is its limited concept hierarchy, where each concept has only very few subconcepts. To alleviate this shortcoming, we added subconcepts to each of the LUBM concepts Course, Department, Professor, and Student by introducing subject areas, such as MathCourse, BioCourse, and CSCourse for courses, MathProfessor, BioProfessor for professors, etc.

We call the resulting TBox LUBM$_n^\exists$ with n indicating the number of subconcepts introduced in Point 2 above (20 by default). For example, LUBM$_{20}^\exists$ contains 106 concept names and 27 role names.

To generate ABoxes, we use the LUBM Data Generator (UBA) version 1.7, modified so as to complement our modifications to the TBox. Specifically, the original UBA generates data that is complete w.r.t. existential restrictions in the LUBM ontology: it produces ABoxes $\mathcal{A}$ such that for every assertion $A(a) \in \mathcal{A}$ and CI $A \sqsubseteq \exists R$ (and $A \sqsubseteq \exists R.B$) in LUBM$_n^\exists$, there is already an R-successor of a in $\mathcal{A}$. Our modifications introduce a controlled amount of incompleteness: the modified data generator takes a probability p as a parameter and, in selected parts of the data, drops generated role assertions with probability p. More information can be found in the appendix of the full version. The second modification of the data generator is linked to the subconcepts introduced in Point 2 above. Whenever the original generator produces an instance a of Student, the new generator randomly chooses a value between 1 and n and generates an assertion for the i-th subject, Subj$i$Student(a); similarly for Course, Department, and Professor.

We use the six queries in Figure 4 that we have hand-crafted specifically for our experiments. Note that cq$_3$ is designed to stress-test the filtering approach: based on the data generation scheme, it is expected to produce a very large number of spurious answers. Requiem test queries are commonly used for benchmarking

query rewriting systems [12]. We did not include those queries since they are too simple for our purposes. In fact, they are answered effortlessly both by our approach and by the query rewriting approach.

5.2 Results

We report on two experiments: in the first experiment we vary the complexity of the ontology by increasing the number of subclasses (the parameter n of the ontology $\text{LUBM}_n^{\exists}$) and in the second experiment we vary the data size by increasing the number of universities that are generated by the modified LUBM data generator. It turned out that, in general, the degree of incompleteness had only very limited effect on the execution time of queries. We therefore do not vary the degree of incompleteness but use 5% incompleteness in the data for both experiments. All experiments were carried out on a Linux (3.2.0) machine with a 3.5Ghz quad-core processor and 8GB of RAM, using IBM DB2 Express-C version 9.7.5.

```
cq1(x,z)<-Student(x), takesCourse(x,y), Subj1Course(y), teacherOf(z,y),
          Professor(z), headOf(z,w), Department(w), memberOf(x,w)
cq2(x)  <-Faculty(x), degreeFrom(x,y), University(y),
          subOrganizationOf(z,y), Department(z), memberOf(x,z)
cq3(x,y)<-Professor(z), memberOf(z,x), Subj3Department(x),
          publicationAuthor(w,z), Professor(v), memberOf(v,y),
          Subj4Department(y), publicationAuthor(w,v)
cq4(x,y)<-Department(x), memberOf(z,x), Student(z), takesCourse(z,v),
          teacherOf(w,v), Professor(w), memberOf(w,y), Department(y)
cq5(x)  <-Person(x), worksFor(x,y), Department(y), takesCourse(x,z),
          Course(z)
cq6(x)  <-Student(x), publicationAuthor(y,x), Publication(y),
          teachingAssistantOf(x,z), Course(z)
```

Fig. 4. Queries cq_1 to cq_6

		Original ABox			Data Completion		
#Univ.	individuals	concepts	roles	time	concepts	roles	time
200	4M	7M	12M	7m30s	12M	22M	16m55s
500	10M	17M	31M	39m06s	31M	55M	70m48s
1000	21M	35M	63M	43m17s	63M	111M	146m39s

Fig. 5. Size original and completed ABox (in million) and load and completion time

The size of the test data for the experiments is detailed in Figure 5, where we give (for 20 subclasses) the number of individuals in the original ABox (there are only about 200 additional individuals in the completion), the number of concept and role assertions (in the original ABox and in its completion), and the load time for the original and the completed ABox (including the completion time).

Test	System	cq_1	cq_2	cq_3	cq_4	cq_5	cq_6
1000.10	Rap-DLog	23.54	TO	40.88	TO	TO	50.41
	Pre-DLog	23.61	TO	43.66	75.33	TO	15.69
	Combo	24.42	393.97	TO	267.55	23.67	38.93
1000.20	Rap-DLog	TO	TO	33.80	TO	TO	54.12
	Pre-DLog	22.86	TO	32.48	TO	TO	17.24
	Combo	18.13	460.16	587.76	266.97	28.30	38.72
1000.40	Rap-DLog	TO	TO	76.69	TO	TC	65.45
	Pre-DLog	23.21	TO	75.71	TO	TO	18.27
	Combo	13.56	456.84	279.09	270.12	28.31	37.43
1000.80	Rap-DLog	TO	TO	UM	TC	TC	TC
	Pre-DLog	TO	TO	UM	TC	TC	17.81
	Combo	7.10	448.69	152.55	268.86	28.07	39.26

Fig. 6. Run time for varying number of subclasses

Test	System	cq_1	cq_2	cq_3	cq_4	cq_5	cq_6
200.20	Rap-DLog	6.72	4.68	3.35	13.37	14.36	9.22
	Pre-DLog	5.84	55.55	5.79	146,26	11.76	2.78
	Combo	4.75	22.65	25.58	51.17	6.50	4.07
500.20	Rap-DLog	14.32	343.47	15.04	TO	TO	25.58
	Pre-DLog	11.32	344.36	14.96	TO	TO	8.18
	Combo	14.14	116.34	161.16	135.36	11.59	19.46
1000.20	Rap-DLog	TO	TO	33.80	TO	TO	54.12
	Pre-DLog	22.86	TO	32.48	TO	TO	17.24
	Combo	18.13	460.16	587.76	266.97	28.30	38.72

Fig. 7. Run time for varying number of universities

	cq_1	cq_2	cq_3	cq_4	cq_5	cq_6
Rap-UCQ	57984	15120	14880	162288	1950	1702
Rap-DLog	85	68	39	81	118	105
Pre-UCQ	TO	15120	14880	TO	1950	1702
Pre-DLog	85	68	39	81	86	63

Fig. 8. Number of disjuncts in UCQ and rules in Datalog program

For our experiments, summarized in Figures 6 and 7, we report the execution time (in seconds; TO stands for 600s timeout, TC for the DB2 output "The statement is too long or too complex", and UM for the DB2 output "Unexpect maxNumBrunch") for the Datalog rewritings generated by Rapid and Presto (transformed into SQL by unfolding them into positive existential queries) and for the Combo filtering approach. We do not report execution times for any UCQ rewritings because they are excessively large and DB2 fails to execute them in all of our experiments, see Figure 8.[1]

[1] We are not aware of any experimental evaluation of the query execution time of rewritings into non-recursive Datalog. Our experiments show that Datalog rewritings can be significantly more efficient than UCQ rewritings.

	cq_1	cq_2	cq_3	cq_4	cq_5	cq_6
spurious answers	0	2	600K	35K	0	0
valid answers	32K	2K	0	20K	0	465K

Fig. 9. Number of answers for 1000 universities, 20 subclasses

The main outcomes of our experiments are as follows:

1. the COMBO filtering approach is significantly more robust than the rewriting approaches when both the complexity of the ontology and the size of the data increase. We observe only one timeout for the filtering approach (cq_3 for 1000 universities and 10 subclasses) but the rewriting approach eventually fails for all queries with the exception of cq_6.
2. For smaller data sets or for simple class hierarchies (e.g., 200 universities and 20 subclasses), the filtering and rewriting approaches are comparable.
3. In contrast to the rewriting approach, the performance of the filtering approach does not depend significantly on the number of subclasses.[2]

The poor performance of the rewriting approach for complex ontologies and large data is due to the fact that for complex queries (the SQL queries corresponding to the datalog rewriting) the DB2 query optimizer realizes that the use of data structures, such as B-tree indices, becomes imperative and attempts to distribute (index) joins into unions to take advantage of these indices. Such an attempt, however, commonly leads to exhausting the resources available for query optimization (DB2 then aborts by outputting TC or UM).

The poor performance of the filtering approach for query cq_3 is due to the large number of spurious answers, see Table 9 for an overview of the number of valid and spurious answers for all queries. It is possible to avoid this behavior at the cost of slight increase of the size of the canonical model: by duplicating the anonymous parts of the canonical model so that no two (or few) individuals in the original ABox 'share' an anonymous part of the canonical model.

We close by commenting on the data loading and completion times reported in Figure 5. Here the completion time is spent almost exclusively on loading the data into the DBMS: indeed, loading large amounts of data into a relational DBMS can be time consuming since the system needs to build up indexes and other auxiliary data structures. Note, however, that bulk-loading data is rare in most applications: standard workloads typically add and remove few tuples at a time. In our case adding (removing), e.g., 100 concept/role assertions into (from) the original ABox results in adding and removing less than 500 tuples from the completed data (for the LUBM ontology), yielding an essentially instant update. Moreover, the changes to the completed data can be efficiently computed given the original ABox and the update request using incremental view maintenance technology [6].

[2] The increase of performance for cq_1 and cq_3 when the number of subclasses grows is due to each subclass becoming less populated as the data size is fixed.

6 Conclusion

We have modified the combined approach to OBDA by replacing the query rewriting part with a filtering technique. This is natural from an implementation perspective and allows to avoid an exponential blowup of the query. We have implemented our approach in the Combo system and generated an improved version of the LUBM benchmark that aims at evaluating OBDA approaches. Our experiments demonstrate the scalability and robustness of our approach.

In the future we plan to extend the combined approach with filtering to other description logics for which, until now, it is unknown how to avoid an exponential blowup of the query. For example, we believe that polytime filtering is possible for the extension of $\mathcal{EL}$ with transitive roles, as found in the OWL2 EL profile. Note that, based on the workshop predecessor [10] of this paper, the combined approach with filtering has already been picked up to implement OBDA for an extension of $\mathcal{EL}$ (but without transitive roles) [16].

From an applied perspective, it would be interesting to compare our approach also with the promising new optimization techniques that have recently been developed in [13,14]. While some of them (such as the exploitation of ABox integrity constraints) aim specifically at the query rewriting approach, others (such as semantic indexing) can easily be used also for the combined approach. We did not include those optimizations and systems in our evaluation because all available implementations seem to require prerequisites that are not satisfied in our tests (such as the availability of mappings from a relational database to the ontology or the storage of the ABox in an in-memory database).

References

1. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P.F. (eds.): The Description Logic Handbook. Cambridge University Press (2003)
2. Calì, A., Gottlob, G., Pieris, A.: New expressive languages for ontological query answering. In: AAAI, pp. 1541–1546 (2011)
3. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Tractable reasoning and efficient query answering in description logics: The DL-Lite family. Journal of Automated Reasoning 39(3), 385–429 (2007)
4. Chortaras, A., Trivela, D., Stamou, G.B.: Optimized query rewriting for OWL 2 QL. In: Bjørner, N., Sofronie-Stokkermans, V. (eds.) CADE 2011. LNCS, vol. 6803, pp. 192–206. Springer, Heidelberg (2011)
5. Eiter, T., Ortiz, M., Simkus, M., Tran, T.K., Xiao, G.: Towards practical query answering for Horn-$\mathcal{SHIQ}$. In: DL, pp. 158–168 (2012)
6. Griffin, T., Libkin, L.: Incremental maintenance of views with duplicates. In: ICMD, pp. 328–339 (1995)
7. Guo, Y., Pan, Z., Heflin, J.: LUBM: A benchmark for OWL knowledge base systems. Web Semantics: Science, Services and Agents on the World Wide Web 3(2-3), 158–182 (2005)
8. Kikot, S., Kontchakov, R., Podolskii, V., Zakharyaschev, M.: Exponential lower bounds and separation for query rewriting. In: Czumaj, A., Mehlhorn, K., Pitts, A., Wattenhofer, R. (eds.) ICALP 2012, Part II. LNCS, vol. 7392, pp. 263–274. Springer, Heidelberg (2012)

9. Kontchakov, R., Lutz, C., Toman, D., Wolter, F., Zakharyaschev, M.: The combined approach to query answering in DL-Lite. In: KR, pp. 247–257 (2010)
10. Lutz, C., Seylan, I., Toman, D., Wolter, F.: The combined approach to OBDA: Taming role hierarchies using filters. In: SSWS+HPCSW, pp. 16–31 (2012)
11. Lutz, C., Wolter, F., Toman, D.: Conjunctive query answering in the description logic $\mathcal{EL}$ using a relational database systems. In: IJCAI, pp. 2070–2075 (2009)
12. Pérez-Urbina, H., Horrocks, I., Motik, B.: Efficient query answering for OWL 2. In: Bernstein, A., Karger, D.R., Heath, T., Feigenbaum, L., Maynard, D., Motta, E., Thirunarayan, K. (eds.) ISWC 2009. LNCS, vol. 5823, pp. 489–504. Springer, Heidelberg (2009)
13. Pinto, F.D., Lembo, D., Lenzerini, M., Mancini, R., Poggi, A., Rosati, R., Ruzzi, M., Savo, D.F.: Optimizing query rewriting in ontology-based data access. In: EDBT, pp. 561–572 (2013)
14. Rodriguez-Muro, M., Calvanese, D.: High performance query answering over DL-Lite ontologies. In: KR, pp. 308–318 (2012)
15. Rosati, R., Almatelli, A.: Improving query answering over DL-Lite ontologies. In: KR, pp. 290–300 (2010)
16. Stefanoni, G., Motik, B., Horrocks, I.: Introducing nominals to the combined query answering approaches for $\mathcal{EL}$. In: DL, pp. 962–974 (2013)
17. Thomazo, M., Baget, J.F., Mugnier, M.L., Rudolph, S.: A generic querying algorithm for greedy sets of existential rules. In: KR, pp. 96–106 (2012)

A Snapshot of the OWL Web

Nicolas Matentzoglu, Samantha Bail, and Bijan Parsia

School of Computer Science, University of Manchester, Manchester, UK
{matentzn,bails,bparsia}@cs.man.ac.uk

Abstract. Tool development for and empirical experimentation in OWL ontology engineering require a wide variety of suitable ontologies as input for testing and evaluation purposes and detailed characterisations of real ontologies. Empirical activities often resort to (somewhat arbitrarily) hand curated corpora available on the web, such as the NCBO BioPortal and the TONES Repository, or manually selected sets of well-known ontologies. Findings of surveys and results of benchmarking activities may be biased, even heavily, towards these datasets. Sampling from a large corpus of ontologies, on the other hand, may lead to more representative results. Current large scale repositories and web crawls are mostly uncurated and suffer from duplication, small and (for many purposes) uninteresting ontology files, and contain large numbers of ontology versions, variants, and facets, and therefore do not lend themselves to random sampling. In this paper, we survey ontologies as they exist on the web and describe the creation of a corpus of OWL DL ontologies using strategies such as web crawling, various forms of de-duplications and manual cleaning, which allows random sampling of ontologies for a variety of empirical applications.

Keywords: Ontology Engineering, empirical methods, corpus, OWL.

1 Introduction

Since its standardisation by the W3C in 2004, the Web Ontology Language OWL[1] has become a widely used language for representing ontological knowledge. OWL ontologies are used across a wide spectrum of domains, ranging from chemistry to bio-health informatics and medical data. There exists an increasing amount of tool support for OWL ontologies, such as OWL reasoners, ontology editors, ontology browsers and visualisation tools, as well as numerous approaches to tasks such as ontology mapping, debugging, and modularisation. Testing and evaluation of proposed techniques and tools form an important part of the development process, and while there are some tools that are specifically tailored towards certain ontologies (such as, for example, the Snorocket reasoner[2] which is aimed at classifying the SNOMED CT ontology [15]), most tools are aimed at general OWL ontologies. One of the core decisions required for a sound empirical

[1] http://www.w3.org/TR/owl2-overview/
[2] http://protegewiki.stanford.edu/wiki/Snorocket

H. Alani et al. (Eds.): ISWC 2013, Part I, LNCS 8218, pp. 331–346, 2013.

methodology is the selection of a suitable dataset and some clarity about that choice and its implications. In particular, choice of data set can threaten both the *internal* validity (i.e. whether a found correlation indicates a causal relation) and *external* validity (i.e. the extent to which the result can be generalised).

Current empirical evaluations, such as OWL reasoner benchmarking and studies on the effectiveness of various debugging techniques, often cherry-pick a few example ontologies or sample from ontology repositories such as the NCBO BioPortal.[3] Alternatively, crawlers such as Swoogle [6] have collected huge amounts of semantic documents, contributing a lot to our understanding of the use of semantic web languages, and allowing us to catch a glimpse of the impact that OWL has on the web ontology landscape. While these crawl-based datasets are certainly useful for many purposes, they do not necessarily lend themselves to ontology research as they collect OWL *files* which may not individually correspond to distinct OWL *ontologies*.

In this paper, we characterise the landscape of OWL ontologies found on the web, with a focus on using *collections* of ontologies for OWL tool development and evaluation purposes. We describe the challenges of gathering a large and meaningful corpus of OWL DL ontologies that is suitable for such experimental tasks, which is based on an automated web crawl combined with several filtering steps to identify OWL ontologies based on heuristics, and to take into account duplicates, versions, and facets of ontologies. We discuss the characteristics of the corpus, such as axiom and constructor usage, OWL profiles, and provenance data, and compare it to several other collections of OWL ontologies that are frequently used (or designed for) testing purposes. The purpose of this paper is twofold: first, we provide insights into the landscape of OWL ontologies found on the web nearly a decade after OWL became an official standard. Second, we highlight the issues faced when selecting suitable test corpora in the OWL tool development process and aim to support tool developers in making informed decisions when choosing test collections.

2 Preliminaries and Background

In this section, we will give a very brief introduction to the web ontology language OWL 2 and the OWL 2 profiles. We then discuss the use of OWL ontology collections in empirical evaluations.

2.1 The Web Ontology Language OWL

OWL 2 [3], the latest revision of the Web Ontology Language OWL, comprises two species of different expressivities, namely OWL 2 DL and OWL 2 Full. The underlying formalism of OWL 2 DL is the description logic $\mathcal{SROIQ}(D)$ [10]. While OWL 2 DL has the familiar description logic semantics (*Direct Semantics*), OWL 2 Full[4] has an RDF-based semantics, which is a superset of the OWL

[3] `http://bioportal.bioontology.org/`

[4] `http://www.w3.org/TR/owl2-rdf-based-semantics/`

2 Direct Semantics; OWL reasoners, however, are restricted to ontologies in (a subset of) OWL DL.

There exist three named 'profiles' for OWL 2, which are syntactic subsets of OWL 2 DL that are tailored towards different applications, trading expressivity of the language for efficient reasoning. The *OWL 2 EL* profile is a tractable fragment of OWL 2 which is based on the description logic $\mathcal{EL}^{++}$ [2]. *OWL 2 QL* (Query Language) which is based on the *DL-Lite* family of description logics [1], has been defined for use in applications which focus on query answering over large amounts of instance data. Reasoning systems for ontologies in the *OWL 2 RL* (Rule Language) profile can be implemented using rule-based reasoning engines.

2.2 Datasets Used in Practice

A wide range of empirical ontology research requires access to a somehow 'interesting' set of ontologies as input to experiments. Empirical studies involving OWL tools and techniques frequently make use of existing ontologies and ontology repositories for test and evaluation purposes. In order to put our work into context with empirical OWL research, we will give an overview of some of the curated OWL ontology repositories and large-scale collections that are commonly used for empirical evaluations.

Curated Ontology Repositories. There exists a number of well-known ontology repositories which are frequently used for empirical experimentation. In what follows, we will briefly describe some of the most prominent repositories and their applications in OWL research.

The ***NCBO BioPortal*** is an open repository of biomedical ontologies which invites submissions from OWL researchers. As of April 2013, the repository contains 341 ontologies in various ontology formats including the full set of OBO Foundry[5] ontologies. Due to its ontologies ranging widely in size and complexity, BioPortal has become a popular corpus for testing OWL ontology applications in recent years, such as justification computation [9], reasoner benchmarking [11], and pattern analysis [13].

The ***TONES*** repository is a curated ontology repository which was developed as part of the TONES project as a means of gathering suitable ontologies for testing OWL applications. It contains 219 OWL and OBO ontologies and includes both well-known test ontologies and in-use ontologies, varying strongly in size and complexity. While ontologies are occasionally added to the repository, it can be considered rather static in comparison with frequently updated repositories, such as BioPortal. The TONES ontologies are frequently used for empirical studies, either as a whole [11,16], by (semi-)randomly sampling from the set [12], or as a source of individual ontologies.

Similar to the TONES repository, the ***Oxford ontology library***[6] is a collection of OWL ontologies gathered for the purpose of testing OWL tools.

[5] `http://www.obofoundry.org/`

[6] `http://www.cs.ox.ac.uk/isg/ontologies/`

The library, which was established in late 2012, currently contains 793 ontologies from 24 different sources, including an existing test corpus and several well-known in-use and test ontologies.

The ***Protégé ontology library***[7] is a submission-based collection of ontologies linking to 95 OWL ontologies including some well-known test and in-use ontologies. While it is not used as frequently as the TONES repository (e.g. [17]), it fulfils a similar purpose of offering a selection of ontologies from a variety of domains.

Large-Scale Crawl-Based Repositories. Crawl-based collections containing thousands and millions of files are popular sources of ontologies used in experiments. While the two largest collections, Swoogle and Watson, seem to be no longer under active development, the Billion Triple Challenge dataset is still updated annually.

Swoogle [6] is a crawl-based *semantic web search engine* that was established in 2004. The crawler searches for documents of specific filetypes (e.g. .rdf, .owl), verifies their status as a valid document of that type, and uses heuristics based on the references found in existing files to discover new documents. In April 2013, Swoogle indexed nearly two million documents, and a search for ontologies (i.e. documents which contain at least one defined class or property) that match 'hasFiletype:owl' returned 88,712 results. While Swoogle is an obvious choice for gathering a large number of OWL ontologies for use in empirical studies (e.g. [17,14,16]), it does not have a public API and prevents result scraping in order to reduce server load, which makes it difficult to gain access to all search results. Furthermore, since the content is not filtered beyond removal of duplicate URLs, a random sample from Swoogle is most likely to return a set of small, inexpressive ontologies, or may be heavily biased towards ontologies from certain domains, as we will discuss in detail in the Section 3.2.

Similar to Swoogle, ***Watson*** [4] is a search engine which indexes documents based on a web crawler that targets semantic web documents. Watson uses filtering criteria in order to only include valid (that is, parseable) documents and ranks results according to their semantic *richness*, which is based on properties such as the expressivity of an ontology and the density of its class definitions. In addition to its web interface, Watson also provides APIs which allow users to retrieve lists of search results for a given keyword. At the time of its release, Watson was indexing around 25,500 documents; however, to the best of our knowledge, the service is no longer under active development.

The ***Billion Triple Challenge*** (BTC) dataset is an annually updated large dataset of RDF/XML documents used in the Semantic Web Challenge.[8] The 2011 set which contains 7.411 million RDF/XML documents crawled from the web using various well-known Linked Data applications as seeds, such as DBPedia and Freebase. According to an analysis by Glimm et al. [7], the set contains just over 115,000 documents that contain a the rdfs:subClassOf predicate, which

[7] `http://protegewiki.stanford.edu/wiki/Protege_Ontology_Library`
[8] `http://challenge.semanticweb.org/`

may be considered sufficient to class the document as an ontology. However, the authors identified that the corpus is biased towards several large clusters of documents from the same domain, which is indicated by the relatively small number of domains (109) that these potential ontologies originate from.

3 Gathering a Corpus of OWL DL Ontologies

While hand curated repositories often lack the potential for generalisability of claims, large-scale document collections suffer from a different problem: they typically contain many small and trivial OWL files as well as large numbers of duplicates, which means that a (naive) random sample is likely to introduce a heavy bias towards irrelevant cases for applications such as reasoner benchmarking and ontology profiling. If we want to make claims about *OWL ontologies on the web*, we need a way to obtain a set of *unique* ontologies (at least to some degree). For our corpus, we consider ontologies that are in OWL DL (and not mere RDFS), contain some logical content and are parseable by the OWL API. In this section, we present our approach to addressing this issue by collecting a large amount of documents through web crawling and applying a series of filtering procedures. The focus of our work lies on the filtering steps applied to arrive at a set with a a high density of (relatively) unique OWL DL ontologies. Table 1 shows an overview of the individual steps in the data curation procedure and the numbers of files filtered out in each step.

3.1 Data Collection

The initial set of documents was collected using a standard web crawler with a large seed list of URLs obtained from existing repositories and previous crawls. The sample obtained for this survey is preliminary in the sense that it is the result of only three weeks of downloading and crawling. We expect the results to improve gradually as the crawler collects more data, which also allows us to refine our heuristics for identifying OWL ontologies. The seeds for the crawl were identified as follows:

- 336,414 URLs of potential ontology files obtained directly from a Google search, Swoogle, OBO foundry, Dumontier Labs,[9] and the Protégé Library.
- 43,006 URLs obtained from an experimental crawl in 2011.
- 413 ontologies downloaded from the BioPortal REST API.

The crawler is based on crawler4j,[10] a multi-threaded web crawler implemented in Java. We use a standard crawling strategy (broad and deep seeding, low crawling depth, i.e. 3 levels), searching for files 'typical' extensions, e.g. owl, rdf, obo, owl.xml, and variations of the type owl.txt, owl.zip, etc. Additionally, the crawler tests whether a link it followed might actually be an OWL file by using

[9] `http://dumontierlab.com/`
[10] `http://code.google.com/p/crawler4j/`

a set of syntactic heuristics (e.g. OWL namespace declaration in all its syntactic variants), thus catching those OWL files that do not have a file extension (or a non-standard one). The crawler only identifies potential URLs, which are then passed on to a candidate downloader that attempts to download files in certain intervals. In the short period of time that the crawler was running, 68,060 new candidate documents where discovered. A large number of candidates in the seeds were not retrievable, due to, amongst others, unreachable domains or possibly restrictions for crawler access.

3.2 Data Curation

Identifying Valid OWL Files. Many surveys of documents on the web acknowledge the necessity of preprocessing crawl results in order to remove irrelevant and duplicate files. Our pipeline for identifying candidate OWL files from the files gathered by the crawler is as follows:

1. Attempting to load and parse files with the OWL API can be computationally expensive, especially for non-OWL files for which the API tries out every possible parser before failing, and for large OWL files. Thus, we applied syntactic heuristics to filter out documents
   - that were clearly not OWL (less than six lines of text, first fifty lines contain the `<html>` tag),
   - or did not contain any OWL declaration (in any syntax) or OBO format version in the first sixty lines.

   This step reduced the initial dataset from 268,944 files to 231,839. A random (statistically significant) sample of 1,037 files that we attempted to load with the OWL API revealed that approximately 11% of the thus removed files were falsely identified as not being OWL.
2. The next step was the removal of byte-identical files. We used Apache Commons IO[11] to determine file stream identity. 43,515 files were grouped into clusters of byte-identical files.
3. Next, all remaining unique files were loaded and saved with the OWL API [8]. Relatively few files (4,590) where not loadable due to parser errors, while 31 did not terminate loading in practical time. After this step, the corpus contained 213,462 valid OWL files.
4. We then removed further duplicates by excluding 6,142 files that have a byte-identical OWL/XML serialisation. The result of the curation pipeline to this point is a set of 207,230 unique (in terms of byte-identical duplicates) and valid OWL files.

Note that we consider the loss of ontologies which cannot be parsed by the OWL API to be negligible, since this API is the most comprehensive of its kind, covering most types of OWL syntaxes and all OWL 2 constructs.

Cluster Detection. One of the main difficulties of gathering a corpus of *ontologies* rather than a corpus of arbitrary OWL files is the problem of identifying

[11] `http://commons.apache.org/io/`

Table 1. Summary of the curation pipeline

Document state	Removed	Size after
Retrieved		268,933
Passed heuristic	37,094	231,839
Passed OWL API, de-duplicated (byte identity)	18,377	213,462
De-duplicated (byte identity after common serialisation)	6,142	207,320
Systematically manually filtered	197,449	9,871
Non-OWL 2 DL and empty ontologies filtered	5,324	4,547

what exactly constitutes a single ontology. This results from the different non-standard ways of publishing ontologies:

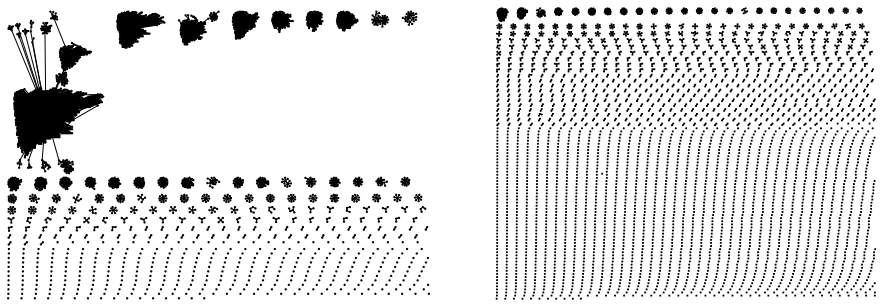
1. There may exist several different *versions* of an ontology. These can be either subsequent versions which have been released in sequence (e.g. version 1.0, 1.1, ...), or slightly modified *variants*, such as 'light' or 'full'.
2. Single ontologies may be distributed over multiple files (e.g. DBPedia, Semantic Media Wikis) or published in modules contained in individual files (*faceted* publishing). The individual files are often very small and describe only trivial fragments of larger OWL ontologies.

In order to identify clusters of versions, variants, and distributed ontologies, we applied two filtering steps based on similar file sizes and file names, and based on the source of the OWL file.

File Name and File Size Patterns. First, a random sample of 100 ontologies was repeatedly drawn from the corpus, and grouped by file size and file name patterns in order to identify clusters of files. If an identified cluster contained large groups of very similar files (such as pages of a Semantic Media Wiki or proofs from Inference Web), all files belonging to the cluster (based on the domain and file name pattern) were removed from the corpus. This process was repeated until a random sample of 100 ontologies appeared heterogeneous enough, i.e. did not contain large numbers of files with obviously similar file names and sizes. In this process, the sample was reduced from 207,230 to just above 19,000 files, which is a reduction by more than 90%.

Domain Names. The file based cluster detection worked well for weeding out the most prominent clusters of distributed ontologies. We then grouped the remaining files by the domain source and inspected the biggest clusters of domains manually to remove files that have no usage (including mere usage of owl:sameAs). Some large contributor domains were eliminated almost entirely (productontology.com), others required more careful attention (sweet.jpl.nasa.gov, for example, provides subsequent versions of each ontology, of which we decided to keep the latest ones).

The largest clusters identified in the cluster detection stage were data generated by various Semantic Media Wikis (146,866), files containing formulas, rules, and related metadata from the Inference Web (19,042). Other notable clusters were generated by the New York Times subject headings SKOS



(a) Before manual cluster removal (b) After manual cluster removal

Fig. 1. Similarity graphs before (sample) and after manual cluster removal

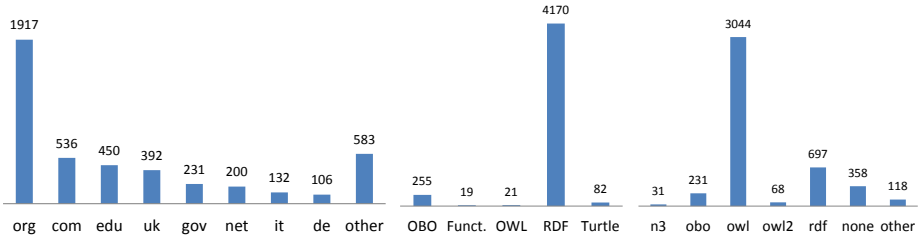
vocabulary (10,438), the UniProt Protein Knowledge Base (5,580 RDF files), as well as files describing instance data of the well-known Friend of a Friend (FOAF) vocabulary (2,312). In total, the clustering process removed over 50% of the ontologies in the crawl set, reducing the corpus to 9,871 files which we presumed to be largely cluster-free.

In order to illustrate the effects of the manual cluster removal, Figure 1 shows two graphs describing the (pairwise) similarity between the ontologies in the corpus before the clustering (on a random sample of 4,547 out of 207,230 ontologies) in Figure 1a and after the clustering (the final 4,547 ontologies, as described in the next section) in Figure 1b. Our notion of similarity is described in section 4.5. We can see that the degree of similarity within the corpus before the filtering is significantly higher than after the cluster removal, with 2,815 connected components before the cluster removal, compared to 521 in the final corpus. Also, the amount of ontologies with no or only few similarity relations is considerably lower after the cluster removal (higher degree of uniqueness).

OWL DL Filtering. Having applied the filtering steps described above, the remaining corpus of OWL ontologies obtained from the crawl contained 9,871 files of which 9,827 files could be loaded.[12] Out of these, 208 were empty (either no axioms, or no entities in the signature, including annotation properties) and 3,207 fell under RDF(S). A further 1,865 ontologies were not in the OWL 2 DL profile for reasons other than missing declarations. We consider missing declarations to be minor violations and thus decided to simply inject them to ensure a more meaningful profile membership (an ontology with a missing class declaration should still be in DL if it was in DL without it).

Apart from missing class- (77.4%), annotation- (67.8%), object property- (34.4%) and data property declarations (15.1%), the main reason for the remaining 1,865 ontologies not falling into OWL DL was the use of reserved vocabulary, most prominently for class IRIs, which occurred in 62.5% of the ontologies and

[12] Since the ontologies were *not* merged with their imports closure at the time of downloading, some ontologies failed loading due to missing imports during the analysis.



(a) Top level domains (b) Syntaxes (c) File extensions

Fig. 2. Provenance data of the ontologies in the crawl corpus

for object properties in 16.5%. Further, a number of ontologies (up to 5%) suffered from various other invalid IRI problems, such as using an IRI for both a datatype and a class (5.2%) and using non-absolute IRIs (2.8%). The remaining issues were caused by the use of non-simple properties in cardinality restrictions. All these violations—which affected almost one fifth of the 9,827 valid OWL files we gathered—cause common OWL DL reasoners to either reject or only process parts of the ontologies.

3.3 Provenance Data

Domain Sources. In the final corpus of 4,547 valid and non-empty OWL files, we count 728 distinct domains (an average of 6.5 ontologies per domain), spread across 52 top level domains. The distribution of top level domains is very similar to the one determined by a Swoogle study characterising the semantic web in 2006 [5]. As Figure 2a shows, '.org' contributes almost half of the documents (42%), followed by '.com' (12%) and '.edu' (10%).

File Extensions and Syntax. Figures 2b and 2c show an overview of the OWL syntaxes and file extensions used for the published OWL files. We can see that the vast majority of ontologies were originally serialised in RDF/XML (4,170), while only a fraction (less than 1%) were published as OWL/XML files. The most frequent file extension used was .owl (67% of the files), followed by .rdf (15%) and .obo (5%). Interestingly, it appears that only a single file in the corpus had the extension .owx, the recommended extension for OWL/XML serialisations.[13]

4 Comparison of OWL Collections

In order to put our crawl-based OWL corpus in context with existing collections of OWL ontologies, we compare its basic ontology metrics against four commonly used datasets. The datasets were selected based on their popularity and intended

[13] `http://www.w3.org/TR/owl2-xml-serialization/`

Table 2. Entity usage (average, median, maximum) in the five collections

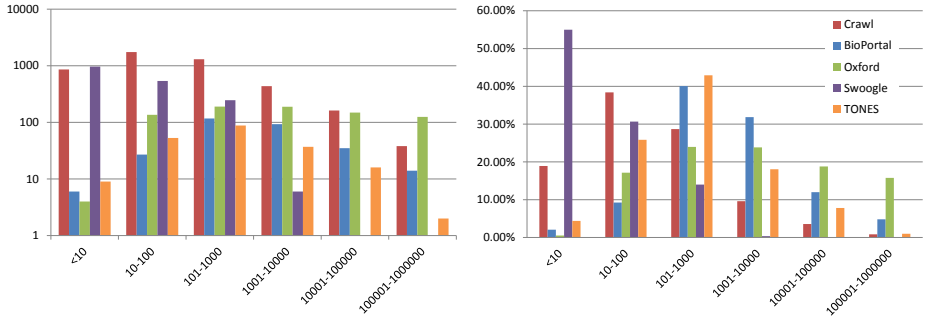
		Crawl	BioPortal	Oxford	Swoogle	TONES
Classes	avg	1,320	11,534	5,652	16	763
	med	27	470	209	9	138
	max	518,196	847,760	244,232	5,104	524,039
Object properties	avg	43	37	43	11	34
	med	8	7	10	15	8
	max	4,951	1,390	964	251	922
Data properties	avg	14	9	5	16	13
	med	1	0	0	18	0
	max	2,501	488	1,371	133	708
Individuals	avg	484	1,075	3,810	29	163
	med	1	0	0	15	0
	max	604,209	232,646	466,937	855	178,308
Logical axioms	avg	3,789	28,050	49,990	60	1,332
	med	69	958	729	8	256
	max	740,559	1,163,895	2,492,725	5,098	1,100,724

use as test corpora, as discussed in Section 2.2; thus, some of the less prevalent sets (e.g. the Protégé library) were excluded. The statistics are given here to allow a comparison between the collections, but no statement is made about which dataset is 'better', as this obviously depends heavily on the purpose. Importantly, the collections in this section are largely left untouched and are *not* curated in the way the Web Crawl was: they may even contain OWL Full and RDFS. The only criterion for inclusion apart from availability was parseability by the OWL API.

The BioPortal and TONES snapshots are from November 2012 and include those OWL and OBO files that could be downloaded and loaded by the OWL API. Files that could be retrieved, but not parsed, usually suffered from unresolvable imports. The third dataset is a sample from a Swoogle snapshot from May 2012 containing OWL and SKOS ontologies. We drew a statistically significant random sample (99% confidence, confidence interval 3) of 1,839 files from the Swoogle snapshot, of which 1,757 could be loaded. The last collection is a snapshot of the Oxford ontology library from April 2013. The final sets were: Crawl (4,547), BioPortal (292), Oxford (793), Swoogle sample (1,757) and TONES (205). For the reasons discussed in section 3.2, missing entity declarations were injected prior to metrics gathering in all cases.

4.1 Entity Usage

Classes, Properties, Individuals. Table 2 shows a detailed overview of the average, median, and maximum values of the relevant logical entities occurring in the five collections (minimum numbers were 0 in all collections, thus they are not listed in the table). Swoogle clearly stands out as a collection with comparatively small numbers of entities per ontology. On average, both the BioPortal and Oxford collections contain very large numbers of logical axioms and classes, with the Oxford collection also containing several ontologies that are particularly



(a) Absolute distribution (b) Relative distributions

Fig. 3. Distribution of ontology sizes, binned by number of logical axioms

heavy on individuals. In comparison, the crawl corpus contains ontologies with on average significantly fewer classes than the curated repositories.

Logical Axioms. In addition to the logical axiom counts given in Table 2, Figures 3a and 3b show a comparison of the ontology sizes in the five collections, sorted into six size bins ranging from less than 10 to over 100,000 logical axioms. We can see that the majority of ontologies in the crawl-based collections (Crawl and Swoogle) are in the lower two bins of fairly small ontologies (less than 100 axioms), whereas the other three collections roughly follow a normal distribution (given this particular binning). On closer inspection we find that the Swoogle snapshot still contains a large number of trivial files from the Semantic Media Wiki, which significantly adds to the number of small ontologies. In the case of the Oxford library and TONES this is likely to be due to the editors explicitly selecting a range of 'interesting' (i.e. medium to large) ontologies.

4.2 Constructors and Axiom Types

Constructors. Figure 4 shows a comparison of the constructor usage in the five collections (as returned by the OWL API). In the crawl corpus, we can see that beyond the basic constructors in $\mathcal{AL}$ (intersection, universal restrictions, existential restrictions of the type $\exists r.\top$, and atomic negation) which are used by the majority (88%) of ontologies in the crawl, property-based constructors, such as inverse properties $\mathcal{I}$ (35% of ontologies) and property hierarchies $\mathcal{H}$ (30%), are the most prevalent across the crawl corpus. Perhaps surprisingly, full existential restriction (of the type $\exists r.C$ for a possibly complex expression C) are only used in 16% of the ontologies. Furthermore, only a very small number of ontologies make use of qualified number restrictions $\mathcal{Q}$ (5%) and complex property hierarchies $\mathcal{R}$ (4%), which might be explained by the fact that they were only introduced with OWL 2.

Regarding the other collections, the Swoogle snapshot only makes use of very few constructors, leaving out most of the more expressive ones.

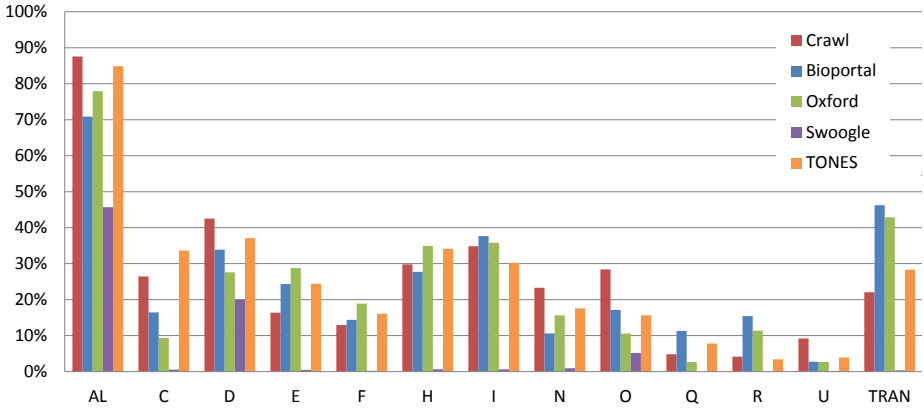


Fig. 4. Frequency of OWL constructor usage in the five collections. Bar height indicates the proportion of ontologies in a collection that use the constructor.

Looking at the axiom type usage in Table 3, this may be explained by the fact that the Swoogle snapshot contains mainly assertion axioms which, in this case, only contain atomic entities and no complex constructors. On the other end of the spectrum, the remaining collections contain similarly large numbers of ontologies using property-related constructors such as transitive properties (TRAN), inverse properties $\mathcal{I}$, and property hierarchies $\mathcal{H}$. While there is no general trend towards a 'most complex' collection, we can find high numbers of ontologies with transitive properties in the BioPortal and Oxford collections, whereas the crawl corpus contains a comparatively large number of ontologies with nominals $\mathcal{O}$ (28%) and unqualified number restrictions $\mathcal{N}$, and all three curated collections (BioPortal, Oxford, and TONES) contain (proportionally) more full existential restrictions $\mathcal{E}$ than the crawl. As with the crawl corpus, the least used constructors in all collections are qualified number restrictions $\mathcal{Q}$ and complex property hierarchies $\mathcal{R}$, along with the union ('or') operator, which occurs in less than 10% of ontologies in all collections.

Axiom Types. Table 3 shows an overview of the most frequent axiom types (in terms of total usage in all collections, not taking into account entity declarations).[14] We can see that by far the most frequently used axiom types in the crawl corpus are AnnotationAssertion and SubClassOf axioms. Domain and range axioms on object properties also occur in nearly half of the ontologies in the corpus; interestingly, their frequency is roughly pairwise identical across all collections, which may indicate that ontology developers generally add domain and range axioms together when introducing object properties. As we have already seen in the discussion on constructors, object property related axiom types such as subproperties, transitive and inverse properties occur frequently in

[14] Note that annotations were removed during the BioPortal download and serialisation process; thus, the corpus does not contain any AnnotationAssertion axioms.

Table 3. Axiom type usage as proportion of ontologies that use an axiom type

	Crawl	BioPortal	Oxford	Swoogle	TONES
SubClassOf	77.0%	96.9%	79.4%	5.6%	92.2%
AnnotationAssertion	78.1%	-	88.9%	36.4%	68.3%
ClassAssertion	44.8%	30.0%	69.7%	35.5%	26.3%
ObjectPropertyRange	47.2%	36.5%	45.8%	1.0%	44.4%
ObjectPropertyDomain	45.6%	38.2%	44.1%	0.9%	43.4%
EquivalentClasses	36.8%	37.9%	44.6%	1.1%	45.4%
SubObjectPropertyOf	30.1%	40.6%	44.4%	0.6%	34.6%
TransitiveObjectProperty	22.0%	46.1%	42.9%	0.3%	28.3%
DisjointClasses	31.0%	42.3%	24.8%	0.3%	41.0%
InverseObjectProperties	31.1%	33.4%	32.8%	0.6%	27.3%
DataPropertyRange	31.5%	29.7%	15.1%	0.6%	27.3%
FunctionalObjectProperty	18.4%	29.4%	24.3%	0.2%	26.8%
DataPropertyDomain	29.6%	27.3%	13.7%	0.5%	22.9%
ObjectPropertyAssertion	17.2%	13.3%	18.5%	26.2%	12.2%
FunctionalDataProperty	15.2%	21.2%	5.4%	0.2%	20.0%
DataPropertyAssertion	13.0%	9.6%	10.8%	19.0%	6.8%

between one fifth and nearly half of the ontologies in the different collections (with the exception of Swoogle). Class related axioms, such as DisjointClasses and EquivalentClasses, can be found equally often in the four collections. This shows that, while the clear majority of axioms are fairly 'trivial' SubClassOf and ClassAssertion axioms, more complex axiom types occur frequently in these OWL ontologies.

4.3 Datatypes

Regarding the usage of datatypes, we found that a very small number of built-in datatypes occur frequently in the five collections, whereas the remaining types are only used rarely. The most frequently used datatypes are rdf:plainLiteral (between 25.9% in BioPortal[15] and 82.1% of the ontologies in the Oxford corpus) and xsd:string datatypes (between 26.8% in the Swoogle snapshot and 59.7% in our crawl corpus). In the Swoogle snapshot, the general datatype usage is lower than in the other collections, with a maximum of only 36.6% of ontologies using rdf:plainLiteral. Interestingly, however, the ontologies in the Swoogle snapshot make more frequent use of xsd:integer (25.3%), xsd:dateTime (25.1% of ontologies), and xsd:decimal (24.2%) than the other collections, which all range between only 1.5% and 10.7% for these types. Finally, across the collections, most other built-in datatypes occur in a small number of ontologies, with the exception of rdfs:literal, which can be found in 18% of the ontologies in the crawl corpus, and xsd:anyURI, which is used in over a third (37.8%) of the ontologies in the Oxford collection.

4.4 OWL Profiles

As mentioned in Section 2, the OWL 2 profiles are relevant for OWL reasoners, which are only compatible with OWL DL ontologies, or may be tailored towards

[15] Due to the removal of annotations in the BioPortal download it is likely that the figures for the BioPortal collection are lower than they would be with annotations.

Table 4. OWL 2 profiles in the collections

	Crawl	BioPortal	Oxford	Swoogle	TONES
Full	0.0%	17.1%	15.3%	3.2%	22.4%
DL	100%	82.9%	84.7%	96.8%	77.6%
EL only	1.8%	16.4%	3.0%	0.0%	9.8%
EL total	4.0%	29.4%	3.2%	0.6%	19.0%
QL only	3.6%	0.7%	0.9%	0.1%	0.0%
QL total	4.8%	33.2%	9.2%	57.5%	18.0%
RL only	15.4%	1.0%	18.0%	33.3%	2.0%
RL total	19.6%	22.9%	27.1%	90.3%	11.7%
DL only	72.7%	29.8%	53.6%	5.8%	46.8%

Table 5. Overlap comparison between the collections

Corpus 1	Corpus 2	Sim.	Con.
Crawl	BioPortal	10.9%	17.6%
Crawl	TONES	11.8%	19.1%
TONES	BioPortal	11.9%	17.9%
Swoogle	Oxford	12.5%	17.3%
Swoogle	BioPortal	14.2%	15.9%
Swoogle	TONES	15.0%	17.1%
Crawl	Swoogle	15.1%	36.3%
Oxford	BioPortal	16.7%	23.2%
Crawl	Oxford	16.8%	24.2%
TONES	Oxford	19.1%	27.0%

a specific subset of OWL 2 DL. Table 4 shows an overview of the profiles for the different ontologies. Note that the profiles are not exclusive, that is, an ontology in one profile may also be in the other profiles; thus, we distinguish between ontologies which are in one profile only, and the total proportion of ontologies in a profile (including other profiles). 'DL only' denotes the proportion of OWL DL ontologies that do not fall into any of the three sub-profiles.

Across the collections, the level of OWL DL ontologies is fairly high (minimum 77.6% in TONES), whereas the occurrence of ontologies in the OWL profiles varies strongly. We can see immediately that the majority of ontologies in the crawl corpus does not fall into any of the sub-profiles EL, QL, or RL, whereas the Swoogle ontologies are largely in a combination of the RL and QL profiles (due to them being fairly inexpressive), with only a fraction (5.8%) being more expressive. A comparatively large number of ontologies (16.4%) in BioPortal fall into the OWL 2 EL (only) profile, which is likely caused by the presence of many large bio-medical ontologies in the corpus that are explicitly designed to be in EL. On the other hand, there are almost no QL or RL only ontologies in BioPortal.

4.5 Overlap Analysis

In order to determine the to which extent the different collections overlap (i.e. shared ontologies), we performed a pairwise comparison of the ontologies in each of the five collections based on two measurements: a) two ontologies are *similar* if the overlap (the intersection of the signatures divided by the union of the signatures) is at least 90%. b) There exists a *containment* relation between two ontologies $\mathcal{O}_1$, $\mathcal{O}_2$, if $sig(\mathcal{O}_1) \subseteq sig(\mathcal{O}_2)$ or $sig(\mathcal{O}_2) \subseteq (\mathcal{O}_1)$. As shown in Table 5, the pairwise similarity overlap (Sim.) between the collections ranges between 10.9% (crawl vs. BioPortal) and 19.1% (TONES vs. Oxford repository). The containment overlap (Con.) between the collections is significantly higher, ranging between 15.9% (Swoogle vs. BioPortal) and 36.3% for the containment relations between the crawl corpus and the Swoogle sample, which is likely to be caused by the heavy use of Swoogle results as seeds for the web crawler.

5 Conclusions and Future Work

In this paper, we have presented an overview of the OWL ontology landscape with a focus on the application of different collections for empirical evaluations. We presented an approach to creating a large yet interesting corpus of OWL DL ontologies suitable for testing and evaluation purposes, characterised the corpus, and compared it to other existing collections of OWL ontologies, such as the NCBO BioPortal and a random sample from the Swoogle search engine. We have seen that drawing a random sample from an unfiltered crawl-based collection may be representative for the general population of OWL files 'found on the web', however, it does not yield relevant data to be used for measuring, for example, reasoner performance on 'actual' ontologies. The direct comparison of these ontology metrics allows OWL tool developers to make an informed decision when selecting a suitable collection of OWL ontologies for testing purposes, while it also shows that a careful filtering procedure of a crawl-based corpus brings the resulting set closer to curated repositories in terms of ontology size and expressivity.

While we believe that we have laid the foundations for a large, crawl-based repository of ontologies for empirical evaluations, we acknowledge some of the limitations our current collection strategy suffers from:

1. Resource limitations (essentially memory allocated to the Java Virtual Machine) might have caused a few very big ontologies to have slipped through in the initial steps of the curation procedure.
2. Web crawlers may not reach the Hidden or Deep Web.
3. The manual curation steps are not easily repeatable.
4. Problems with unavailable ontology imports.

The main limitations of our approach stem from general problems with web crawling, since it is unlikely that we will be able to index *all* OWL ontologies that are reachable on the web. However, we expect that a stronger focus on meta crawling (i.e. crawling search engines) and more extensive (manual) repository reviewing will gradually expand our seed. With the insights we have gained into general cluster characteristics, we aim to replace the manual filtering procedures by automated ones. The problem of unavailable ontology imports can be easily solved by downloading the imports closure of an ontology in the crawling and ontology validation process.

In addition to improving the crawling and validation strategies and the analysis of the actual content of the ontologies, we focus on establishing a repository of OWL ontologies that allows researchers to retrieve specific samples of ontologies for various empirical tasks. One common problem for ontology researchers is the retrieval of a set of ontologies of a particular characteristic, for example *'a set of OWL 2 EL ontologies with more than 100 axioms'*. We plan to provide an infrastructure that makes it possible to retrieve datasets that can also be made permanently accessible to other researchers, thus aiding the reproducibility of empirical experimentation. A prototype of this repository can be found at `http://owl.cs.manchester.ac.uk/owlcorpus`.

Nicolas Matentzoglu is supported by a CDT grant by the UK EPSRC.

References

1. Artale, A., Calvanese, D., Kontchakov, R., Zakharyaschev, M.: The DL-Lite family and relations. J. of Artificial Intelligence Research 36, 1–69 (2009)
2. Baader, F., Brandt, S., Lutz, C.: Pushing the $\mathcal{EL}$ envelope. In: Proc. of IJCAI 2005, pp. 364–369 (2005)
3. Cuenca Grau, B., Horrocks, I., Motik, B., Parsia, B., Patel-Schneider, P.F., Sattler, U.: OWL 2: The next step for OWL. J. of Web Semantics 6, 309–322 (2008)
4. d'Aquin, M., Baldassarre, C., Gridinoc, L., Sabou, M., Angeletou, S., Motta, E.: Watson: supporting next generation semantic web applications. In: Proc. of WWW 2007 (2007)
5. Ding, L., Finin, T.W.: Characterizing the semantic web on the web. In: Cruz, I., Decker, S., Allemang, D., Preist, C., Schwabe, D., Mika, P., Uschold, M., Aroyo, L.M. (eds.) ISWC 2006. LNCS, vol. 4273, pp. 242–257. Springer, Heidelberg (2006)
6. Ding, L., Finin, T., Joshi, A., Pan, R., Cost, R.S., Peng, Y., Reddivari, P., Doshi, V., Sachs, J.: Swoogle: A search and metadata engine for the semantic web. In: Proc. of CIKM 2004 (2004)
7. Glimm, B., Hogan, A., Krötzsch, M., Polleres, A.: OWL: yet to arrive on the web of data? In: Proc. of LDOW 2012 (2012)
8. Horridge, M., Bechhofer, S.: The OWL API: a Java API for OWL ontologies. Semantic Web J. 2(1), 11–21 (2011)
9. Horridge, M., Parsia, B., Sattler, U.: Extracting justifications from BioPortal ontologies. In: Cudré-Mauroux, P., Heflin, J., Sirin, E., Tudorache, T., Euzenat, J., Hauswirth, M., Parreira, J.X., Hendler, J., Schreiber, G., Bernstein, A., Blomqvist, E. (eds.) ISWC 2012, Part II. LNCS, vol. 7650, pp. 287–299. Springer, Heidelberg (2012)
10. Horrocks, I., Kutz, O., Sattler, U.: The even more irresistible $\mathcal{SROIQ}$. In: Proc. of KR 2006 (2006)
11. Kang, Y.-B., Li, Y.-F., Krishnaswamy, S.: Predicting reasoning performance using ontology metrics. In: Cudré-Mauroux, P., Heflin, J., Sirin, E., Tudorache, T., Euzenat, J., Hauswirth, M., Parreira, J.X., Hendler, J., Schreiber, G., Bernstein, A., Blomqvist, E. (eds.) ISWC 2012, Part I. LNCS, vol. 7649, pp. 198–214. Springer, Heidelberg (2012)
12. Keet, C.M.: Detecting and revising flaws in OWL object property expressions. In: ten Teije, A., Völker, J., Handschuh, S., Stuckenschmidt, H., d'Acquin, M., Nikolov, A., Aussenac-Gilles, N., Hernandez, N. (eds.) EKAW 2012. LNCS, vol. 7603, pp. 252–266. Springer, Heidelberg (2012)
13. Mikroyannidi, E., Manaf, N.A.A., Iannone, L., Stevens, R.: Analysing syntactic regularities in ontologies. In: Proc. of OWLED 2012 (2012)
14. Nguyen, T.A.T., Power, R., Piwek, P., Williams, S.: Measuring the understandability of deduction rules for OWL. In: WoDOOM 2012 (2012)
15. Spackman, K.A., Snomed, R.T., Snomed, C.T.: Promise of an int. clinical ontology. M.D. Computing 17(6), 29 (2000)
16. Third, A.: Hidden semantics: what can we learn from the names in an ontology? In: Proc. of INLG, pp. 67–75 (2012)
17. Wang, T.D., Parsia, B., Hendler, J.: A survey of the web ontology landscape. In: Cruz, I., Decker, S., Allemang, D., Preist, C., Schwabe, D., Mika, P., Uschold, M., Aroyo, L.M. (eds.) ISWC 2006. LNCS, vol. 4273, pp. 682–694. Springer, Heidelberg (2006)

Semantic Rule Filtering for Web-Scale Relation Extraction

Andrea Moro[1], Hong Li[2], Sebastian Krause[2],
Feiyu Xu[2], Roberto Navigli[1], and Hans Uszkoreit[2]

[1] Dipartimento di Informatica, Sapienza Università di Roma
Viale Regina Elena 295, 00161 Roma, Italy
{moro,navigli}@di.uniroma1.it
[2] Language Technology Lab, DFKI, Alt-Moabit 91c, Berlin, Germany
{lihong,skrause,feiyu,uszkoreit}@dfki.de

Abstract. Web-scale relation extraction is a means for building and extending large repositories of formalized knowledge. This type of automated knowledge building requires a decent level of precision, which is hard to achieve with automatically acquired rule sets learned from unlabeled data by means of distant or minimal supervision. This paper shows how precision of relation extraction can be considerably improved by employing a wide-coverage, general-purpose lexical semantic network, i.e., BabelNet, for effective semantic rule filtering. We apply Word Sense Disambiguation to the content words of the automatically extracted rules. As a result a set of relation-specific relevant concepts is obtained, and each of these concepts is then used to represent the structured semantics of the corresponding relation. The resulting relation-specific subgraphs of BabelNet are used as semantic filters for estimating the adequacy of the extracted rules. For the seven semantic relations tested here, the semantic filter consistently yields a higher precision at any relative recall value in the high-recall range.

Keywords: Relation Extraction, Semantics, WSD, Rule Filtering, Web-scale, Semantic relations.

1 Introduction

Information Extraction (IE) automatically finds relevant entities or relations (including facts and events) in natural language texts. The task of Relation Extraction (RE) is to recognize and extract instances of semantic relations between entities or concepts mentioned in these texts. Usually the relations are given, but they may also be induced from the data, as in Open IE [3] where tuples of potential relations are extracted without role labeling. In this paper, we address Web-scale domain-adaptive RE with semantic labeling for given relations of varying arity.

Precision and recall are two important performance measurements. In the past, recall, scalability, domain adaptability and efficiency were regarded as much greater challenges than achieving high precision, because research employed learning data limited in size, types and domains that did not give rise to the noise levels encountered when using the Web as learning corpus. Much research also concentrated on intelligence applications, where recall is much more important than precision. But the limited learning

H. Alani et al. (Eds.): ISWC 2013, Part I, LNCS 8218, pp. 347–362, 2013.

data were not sufficient to overcome the recall barriers, because of the long tail in the skewed frequency distribution of relevant linguistic patterns. Today, the availability of (i) large open-source knowledge databases such as Freebase [6], (ii) nearly unlimited textual resources on the Web and (iii) efficient NLP systems such as dependency parsers (e.g., [2,46]) enables the creation of large-scale distantly (or minimally) supervised RE systems for many relations with acceptable efficiency [22,25,36,38,59]. These systems can achieve much better recall without the need for larger volumes of labeled data. Their drawback is their lack of precision, resulting from the large number of candidate patterns which are selected but not sufficiently constrained by the seed knowledge. Filtering by lexical features (e.g., part-of-speech information, word sequences, etc.), syntactic features such as dependency relations, or simple manually-defined heuristics [1,4,8,22,25] does not suffice. A major open challenge is the exploitation of semantic information in the text and in existing semantic resources beyond the seed data. Several recent approaches add secondary semantic features to their systems which, however, have been shown to offer only slight improvements in RE precision [19,60].

In this paper, we propose a new method that automatically learns relation-specific lexical semantic resources from a general-purpose knowledge base without any task-specific manual annotation. The input of this unsupervised learning method is a large collection of noisy RE patterns (40K rules per relation on average) acquired by the RE system Web-DARE [22], together with their sentence mentions from 20M Web pages retrieved by searching for the named-entity tuples of the seed facts. The patterns are dependency structures extracted from the parse trees of the sentence mentions. The learning system acquires relation-relevant word senses by applying Word Sense Disambiguation [30] to the words in the patterns and then extracts the corresponding relation-specific lexical semantic subgraphs from a large-scale general purpose lexical semantic network, i.e., BabelNet [31]. These relation-specific subgraphs are utilized as semantic knowledge for filtering out bad rules. In contrast to frequency-based filters, our semantic rule filter, on the one hand, deletes those high-frequency rules which do not contain any relation-relevant words, but at the same time, on the other hand, it also preserves any low-frequency rules which are semantically relevant (owing to their low frequency such rules would previously have been, erroneously, filtered out). It thereby increases both precision and recall.

The main contributions of this paper are to:

- introduce a novel unsupervised, scalable learning method for automatically building relation-specific lexical semantic graphs representing the semantics of the considered relation. Moreover, we show the usefulness of these graphs for filtering semantically irrelevant rules and improving the precision of large-scale RE;
- report on a first comparison of WordNet and BabelNet with respect to improving RE: BabelNet achieves better recall and F-score than WordNet both in rule filtering and in RE;
- demonstrate that relation-specific lexical semantic resources can improve RE performance: For seven semantic relations tested, the semantic filter consistently yields a higher precision at any relative recall value in the high-recall range.

2 Related Work

In recent years several approaches to RE have tried to circumvent the costly, and still not satisfactory, corpus annotation needed for supervised learning. Minimally or weakly supervised methods start with limited initial knowledge and unlabeled data. By a bootstrapping process partial labeling of data and system training are performed in several iterations (e.g., [1,7,8,39,55]). However, these systems often have to cope with low recall and precision, the latter partially due to semantic drift.

A newer class of approaches, sometimes referred to as distant supervision, utilizes extensive volumes of preexisting knowledge for partially labeling large volumes of data. [25] train a *linear-regression* classifier on *Freebase* relation instances occurring in a large Wikipedia corpus. In order to achieve high precision (without much consideration for recall), lexical features, syntactic dependency information and negative examples are employed. The resulting precision is 67.6% for 10,000 sampled instances of 102 relations.

Open IE systems such as *TextRunner* and its successors [3,4,13,56], together with subsequent developments [48,27,28], detect instance candidates of any unknown relation. The Open IE task, however, is faced with even higher levels of noise. Shallow linguistic analysis and numerous heuristics based on lexical features and frequencies are utilized to filter out noisy or irrelevant information for both learning and extraction.

However, all these RE methods are faced with the problem of estimating the confidence of automatically labeled information and learned rules. Some approaches use the confidence value of the extracted instances or the seed examples as feedback for estimating the confidence of rules [1,7,55]. In most cases, however, the confidence values rely on redundancy. Many approaches utilize negative examples for filtering [25,51,54]. As mentioned above, lexical features such as word sequences or part-of-speech information are often utilized for further filtering [3,4,25,56]. Some approaches employ domain-relevant terms for filtering rules [35,52]. Web-DARE [22] filters rules by their absolute frequency and their relative frequency in comparison to other related relations (overlap). In order to improve precision, NELL – a large-scale RE system designed to learn factual knowledge from the Web in a never-ending manner [8] – employs the "coupled learning" of a collection of classifiers for several relations. By exploiting this method it is possible to filter out noisy relation instances recognized by mutually exclusive classifiers. However, even if some of these approaches reach the goal of high precision, this is obtained at the cost of recall.

To obtain high precision while at the same time preserving recall, the use of semantic approaches can be highly beneficial. One of the first attempts was presented in [24] where the authors proposed a method for adding semantic features to the labeled data used for training a syntactic parser. However, even if the authors obtained promising results, the major drawback of this approach is the need for huge volumes of annotated data, which, even today, is hard to obtain. Other approaches add semantic features to feature-based RE systems that learn relation-specific extractors [20,60]. However, none of these approaches has taken full advantage of syntactic and semantic analysis, and thus they have achieved only small improvements [19]. A recent trend in this research strand is the utilization of tree kernel-based approaches, which can efficiently represent high-dimensional feature spaces [36,59]. However, supervision is stil required and semantic

analysis is only marginally employed. In contrast, in this paper we draw only upon semantic knowledge to obtain significant improvements over non-semantic systems.

To integrate and make the most of semantics in RE systems we need a lexical representation of knowledge that can be exploited to obtain a semantic description of the relations. In contrast to many state-of-the-art resources [15,18,29], BabelNet [31] integrates encyclopedic (i.e., from Wikipedia) and lexicographic knowledge (i.e., from WordNet) to obtain a rich multilingual "encyclopedic dictionary".

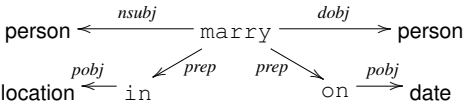
3 Web-DARE and NELL

Our goal is to leverage semantic knowledge to improve the quality of the RE rules learned by two Web-scale RE systems, i.e., Web-DARE and NELL, introduced hereafter.

3.1 Web-DARE

The Web-DARE system [22] learns RE rules for n-ary relations in a distant-supervision manner [25]. For 39 relations, 200k instances, i.e. *seeds*, were collected from the freely-available knowledge base Freebase. Utilizing these relation instances as Web-search queries, a total of 20M Web pages were retrieved and processed, extracting from them 3M sentences mentioning the arguments (entities) of a seed instance. After analyzing these sentences by additional NER and parsing, 1.5M RE rules were extracted from the dependency parses. The following example rule contains four arguments, two married persons plus the wedding location and the starting date of the marriage:

(1)



FO-Filter. The reported recall for Web-DARE is rather high. To overcome the extremely low precision, a rule filter (called FO-Filter) is introduced based on the rule frequency and mutual exclusiveness of relations with similar entity-type signatures. Whenever a particular rule has been learned for more than one relation, it will be added to one relation if its relative frequency in this relation is the highest in comparison to other relations. Rule frequency is the number of the sentence mentions from which a rule has been learned. Relative frequency of a rule in a relation is calculated on the basis of the frequency of this rule in this relation compared to the total frequency of all rules in this relation. Furthermore, a frequency threshold has been applied to exclude rules with low frequency.

3.2 NELL

NELL [8] is a system designed to learn factual knowledge from an immense corpus over a long period. NELL's background ontology contains several hundred entity types (categories) and binary relations, which are related in that certain pairs of categories or relations are marked as being sub- or supersets of each other, or as being mutually

exclusive. This coupling of relations is beneficial when estimating the correctness of newly extracted facts. Earlier versions of NELL, described by [5] and [9], were based mainly on a learner of lexico-syntactic rules. The architecture was extended with an extractor working on semi-structured parts of Web pages, i.e., HTML lists and tables, by [10]. Afterwards, a classifier for categorizing noun phrases into entity types based on morphological features as well as an inference-rule learning component was added to NELL [8,23]. NELL's rules are binary and surface-level oriented, as illustrated by the following example:

(1) person `and husband` person.

While in the NELL system these patterns are only a single piece in a bigger learning and extraction pipeline, we employ them here *on their own* for RE. The NELL rules serve mainly as an additional testing ground for our semantic filter. This implies that the RE results presented in Section 6 are not representative of the performance of NELL itself.

4 WordNet and BabelNet

In this section we give a brief overview of the knowledge bases that we use to obtain a semantic description of the considered relations. The first is WordNet [15] which is a manually-created lexical network of the English language, initiated by George A. Miller in the mid-1980s. The two main components of this resource are the synsets and the semantic relations between them. A synset is a set of synonyms representing the same concept. Each synset is connected to other synsets through lexical and semantic relations. There are roughly 20 relations, among which are hyponymy, meronymy and entailment.

The second resource that we draw upon is BabelNet[1] [31], a large-scale multilingual semantic network which, in contrast to WordNet, was built automatically through the algorithmic integration of Wikipedia and WordNet. Its core components are the Babel synsets, which are sets of multilingual synonyms. Each Babel synset is related to other Babel synsets by semantic relations such as hypernymy, meronymy and semantic relatedness, obtained from WordNet and Wikipedia. Moreover, since BabelNet is the result of the integration of a lexical resource and an encyclopedic resource, it is perfectly in line with the multilingual linguistic Linked Open Data project [12]. This project consists of a vision of the Semantic Web in which a wealth of linguistic resources are linked to each other so as to obtain a bigger and optimal representation of knowledge [32].

One major difference between these two resources is in respect of their considerably different sizes, both in terms of number of concepts and semantic relation instances. On the one hand, WordNet provides roughly 100K synsets, 150K lexicalizations and 300K relation instances. On the other hand, BabelNet contains roughly 5.5M synsets, 15M lexicalizations and 140M relation instances. Moreover, given the multilingual nature of BabelNet (the current version 1.1.1 considers six different languages: Catalan, English, French, German, Italian and Spanish), this resource can exploit multilinguality to perform state-of-the-art knowledge-based Word Sense Disambiguation [33] (in contrast to WordNet which encodes only English lexicalizations), thereby enabling new methods for the automatic understanding of the multilingual (Semantic) Web.

[1] `http://babelnet.org`

5 Rule Filtering with Relation-Specific Semantic Graphs

Current statistical approaches to the rule filtering problem do not take into account the semantic information available within the rules. As a consequence they are not able to identify bad rules, which, from the point of view of the extracted arguments, look correct. For instance, the rule *PERSON met PERSON*, extracted for the relation *married*, is not specific to the considered semantic relation even if it extracts several good relation instances. We tackle this issue by introducing a novel approach to explicitly represent the semantics of each rule and relation. To do this, we apply Word Sense Disambiguation (WSD) to the automatically extracted rules and then build relation-specific semantic graphs which represent the semantics of the considered relation. For instance, our semantic representation of the relation *married* contains concepts that are semantically distant from the concepts usually associated with the term *met*. As a result, our approach is able to correctly filter out the aforementioned rule.

5.1 Building Semantic Graphs

Given a semantic relation ρ, we consider the set of rules R_ρ automatically extracted by the Web-DARE system, together with the set S of sentences from which these rules were extracted. Our goal is to build a semantic representation for the relation ρ. In Algorithm 1 we show the pseudocode of our semantic graph construction approach, described in the following.

WSD (lines 4–13 of Algorithm 1). In this first part of the algorithm we compute a frequency distribution over the synsets of the considered knowledge base for the semantic relation ρ. Given the set S of sentences used by the Web-DARE system and a rule $r \in R_\rho$, we define the subset $S_r \subset S$ as the set of sentences from which the rule r was extracted (see line 6). For instance, we add the sentence *It was here that the beautiful Etta Place first met Harry Longabaugh* to the set $S_{PERSON\_met\_PERSON}$. Then, for each sentence s in S_r, we perform WSD on each content word of the rule r using the remaining content words of s as context (see line 9). For instance, using the previous sentence and given the word *met*, we use as context the following words: *was, here, beautiful, Etta, Place, first, Harry, Longabaugh* obtaining the synset[2] $meet^1_v$. We use an off-the-shelf API for knowledge-based WSD[3] which exploits a knowledge base and graph connectivity measures to disambiguate words [34]. For each synset selected by the WSD API, we increment its count (see line 10 in Algorithm 1). As a result of this step, we obtain Σ_ρ, a synset frequency distribution representing the unstructured semantics of the given relation ρ (see lines 4–10 in Algorithm 1). Then, to avoid data sparsity, we discard all the synsets that occur only once (lines 11–13). For example, given the semantic relation $\rho = marriage$, the most frequent synsets returned by the WSD API are: $marry^1_v$, $wife^1_n$ and $husband^1_n$.

[2] For ease of readability, in what follows we use senses to denote the corresponding synsets. We follow [30] and denote with w^i_p the i-th sense of w with part of speech p.

[3] We did not use supervised approaches as they would have required a separated training phase for each considered domain.

Algorithm 1. Building the relation-specific semantic graph

input: S, the set of sentences from which the rules where extracted;
R_ρ, the set of rules automatically extracted for the the relation ρ;
E_{kb}, the edges, i.e. the semantic relation instances, of the knowledge base;
k, our free parameter.
output: G_ρ, the semantic graph for ρ.
function SEMANTICGRAPH(S, R_ρ, E_{kb}, k)
    Σ_ρ := Map<Synset, Integer>
    for each $r \in R_\rho$ **do**
        $S_r := \{s \in S : s \text{ matches } r\}$
        for each *sentence* $\in S_r$ **do**
            for each *word* $\in$ contentWords(r) **do**
                synset := WSD(*word, sentence*)
                Σ_ρ[*synset*]++ // we increase by one the integer associated with the synset
    for each *synset* $\in$ *keys*(Σ_ρ) **do**
        if Σ_ρ[*synset*] $= 1$ **then**
            Σ_ρ.*remove*(*synset*)
    $\Gamma_\rho := Top(\Sigma_\rho, k)$ // we initialize the core synsets with the top-k most frequent synsets
    for each *synset* $\in$ *keys*(Σ_ρ) **do**
        if $\exists$ *synset*$'$ $\in Top(\Sigma_\rho, k)$ s.t. (*synset*, *synset*$'$) $\in E_{kb}$ **then**
            $\Gamma_\rho := \Gamma_\rho \cup \{synset\}$
    return $G_\rho := (\Gamma_\rho, \{(synset_1, synset_2) \in E_{kb} : synset_1, synset_2 \in \Gamma_\rho\})$

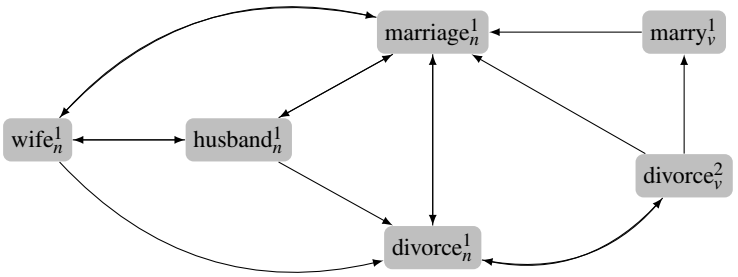


Fig. 1. An excerpt of the semantic graph associated with the relation *marriage* with $k = 2$

Core Synsets (lines 14–18 of Algorithm 1). In the second part of Algorithm 1 we build a subset $\Gamma_\rho \subseteq \Sigma_\rho$ of *core synsets*, i.e., the most semantically representative concepts for a semantic relation ρ. We initialize Γ_ρ with the top-k most frequent synsets in Σ_ρ (line 14). For instance, with $k = 2$ and the relation $\rho = marriage$, we have $\Gamma_{marriage} := \{marry_v^1, wife_n^1\}$. We then look at each synset s in Σ_ρ and we check if there exists a semantic relation in the knowledge base that connects the synset s to any of the top-k frequent synsets. If this is the case, we augment Γ_ρ with s, i.e., we extend our initial set of core synsets with additional semantically related synsets (lines 15–17). For example, with $k = 2$ and the relation $\rho = marriage$, we add $husband_n^1$, $marriage_n^1$ and $divorce_v^2$ to $\Gamma_{marriage}$, among others (see Figure 1). Finally, the algorithm returns the subgraph G_ρ

Algorithm 2. Classifying the rules of a semantic relation

```
1: input: G_ρ, the semantic graph associated with the relation ρ;
          R_ρ, the set of rules associated with the relation ρ;
2: output: GR, the good rules
3: function FILTER(G_ρ, R_ρ)
4:     GR := ∅
5:     for each rule ∈ R_ρ do
6:         if ∃ word ∈ contentWords(rule), synset ∈ V(G_ρ) such that
             word ∈ lexicalizations(synset) then
7:             GR := GR ∪ {rule};
8:     return GR;
```

of the given knowledge base induced by the set of core synsets Γ_ρ (see line 18), which will be used to filter out bad rules as described in Section 5.2. An excerpt of the kind of graphs that we obtain is shown in Figure 1.

5.2 Filtering Out Bad Rules

We now describe our semantic filter, whose pseudocode is shown in Algorithm 2, which filters out bad rules by exploiting the semantic graph G_ρ previously described. For each rule $r \in R_\rho$ associated with a semantic relation ρ, we check if any of its content words matches one lexicalization of the concepts contained in the semantic graph G_ρ (see line 6). If this is the case we mark r as a good rule, otherwise we filter out r. For instance, our filter recognizes the rule *PERSON married PERSON* as a good rule, while filtering out *PERSON met PERSON* because none of the senses of $meet_v$ matches any of the core synsets automatically associated with the relation *married*.

6 Experiments and Evaluations

6.1 Experimental Setup

Overview. We carried out two different experiments to assess the quality of our semantic filtering algorithm: an intrinsic evaluation (i.e., evaluating the quality of the filtered rules against a gold-standard rule set without taking into account the extraction performance) and an extrinsic evaluation (i.e., determining its effect on recall and precision of real RE). In both evaluations, we experiment with different values of k for Algorithm 1, ranging from 1 to 15.

Our evaluation aims at obtaining insights concerning the following aspects:

- *rule-frequency driven FO Filter vs. filtering based on lexical semantics:* We test our semantic rule filter against the previous FO-Filter to compare the performance difference.
- *impact of the selection among lexical semantic resources:* We evaluate the effects of training our filtering algorithm with two different knowledge bases: manually generated WordNet vs. BabelNet, a massive extension of WordNet automatically created from Wikipedia information (cf. Section 4).

Table 1. Statistics about (a) the input data for the rule filters, (b) the gold standard for intrinsic evaluation, (c) the baseline (pre-filtering) performance for the extrinsic evaluation. Values are shown for both Web-DARE (WD) and NELL (N) systems. "Freebase Mentions" refers to the number of correctly identified Freebase mentions in a sample of the evaluation corpus.

Relation	INPUT		INTRINSIC (SEC. 6.2)	EXTRINSIC (SEC. 6.3)						
	# Rules		# Gold-Set Rules	# Extracted Mentions		Baseline Precision		# Freebase Mentions		
	WD	N	WD (+|-)	WD	N	WD	N	WD	N	WD ∪ N
acquisition	26,986	272	52|48	17,913	296	14.20%	28.04%	93	1	93
marriage	88,350	547	47|53	92,780	2,586	11.60%	8.50%	161	9	168
person birth	22,377	995	50|50	63,819	2,607	36.50%	5.60%	77	0	77
person death	31,559	5	50|50	84,739	17	18.00%	100.00%	300	0	300
person parent	45,093	956	22|78	93,800	358	13.20%	66.20%	91	5	92
place lived	47,689	829	51|49	84,389	3,155	47.90%	92.00%	68	38	106
sibling relationship	26,250	432	12|88	59,465	211	5.60%	51.18%	48	2	49
sum	288,304	4,036	284|416	496,905	9,230	–	–	838	55	885
average	41,186	577	41|59	70,986	1,319	21.00%	50.22%	120	20	126

– *generality of the semantic filtering method:* We also apply our filtering to the NELL rule set to check whether the filtering is general enough to apply beyond DARE rules.

Table 1 lists our target relations in column "Relation" while in column "INPUT" we show the respective number of rules given by the Web-DARE and NELL[4] systems.

6.2 Intrinsic Evaluation

Dataset. For the intrinsic evaluation, we manually validate a set of 700 Web-DARE rules to create a balanced gold standard of correct rules (+) and incorrect ones (-) from all target relations. Column "INTRINSIC" of Table 1 presents the number of manually validated rules per relation.

Results. In this section we describe the intrinsic evaluation of our filtering algorithm. To evaluate the filtered rules, we compute their precision, recall and F-score against the manually built gold standard rule set. We do this without considering the relation extraction performance of the filtered rules, i.e., how many good relation instances are effectively extracted by these rules, as this is the focus of the extrinsic evaluation.

Figure 2 displays precision, recall and F-score values for the total set of seven semantic relations using WordNet and BabelNet as knowledge bases and varying the parameter k from 1 to 15 (see Algorithm 1 in Section 5). As Figure 2 shows, we obtain a considerable increase in recall by using BabelNet instead of WordNet (with a maximum value of roughly 90% for BabelNet and 70% for WordNet). Despite the gain in recall

[4] NELL rules were taken from iteration 680, `http://rtw.ml.cmu.edu/resources/results/08m/`

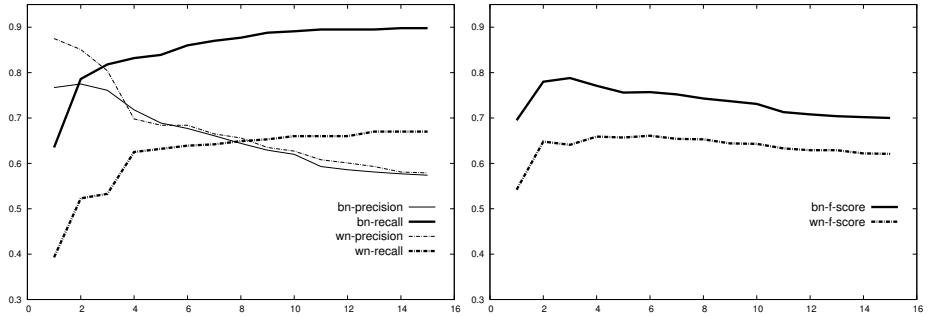


Fig. 2. Precision, Recall and F-score considering all the 7 semantic relations, using WordNet (dotted) and BabelNet (solid), varying the value k from 1 to 15

for the BabelNet filter, precision stays roughly the same as for the WordNet filter (for each value of k), which yields an F-score boost of roughly 10%.

The main reason for the observed improvement can be found in the rich set of semantic relation instances of BabelNet, i.e. when using BabelNet as our knowledge base, the filtering method is able to discover semantic connections between concepts that are not provided by WordNet. For instance, WordNet does not contain a semantic connection between the concepts of *marriage* and *divorce*, whereas BabelNet does.

6.3 Extrinsic Evaluation

Dataset. In the extrinsic evaluation, we use the Los Angeles Times/Washington Post (henceforth LTW) portion of the English Gigaword v5 corpus [37] for RE. LTW is comprised of 400K newswire documents from the period 1994–2009. We match all Web-DARE and NELL rules against the LTW corpus, resulting in more than 500K detected relation mentions, shown in column "EXTRINSIC" of Table 1. To estimate the precision of RE, we manually check a random sample of 1K extracted mentions per relation and system, giving us the pre-filtering performance depicted in column "Baseline Precision". To estimate the RE coverage of the rules, we investigate how many mentions of Freebase facts the systems find on LTW. The values are listed in the last three columns of Table 1, labeled "Freebase Mentions". Only actual mentions are taken into account, i.e., sentences containing the entities of a Freebase fact and actually referring to the corresponding target relation. Relative recall values stated in this section are to be understood as recall with respect to the set of Freebase-fact mentions found by at least one of the two rule sets (Web-DARE/NELL), i.e., relative to the very last column of Table 1.

Semantic Filter for Web-DARE Rules. Figure 3 presents the precision vs. relative recall results of RE when performed with the baseline Web-DARE rules, the statistical approach (FO-Filter) and our semantic filtering algorithm (S-Filter) using BabelNet and WordNet. The FO-filter is able to increase the precision from the baseline of 20% up to close to 100%, since by varying the frequency threshold, any value between 40% and 98% can be reached. But this filtering sacrifices a large portion of the initial recall.

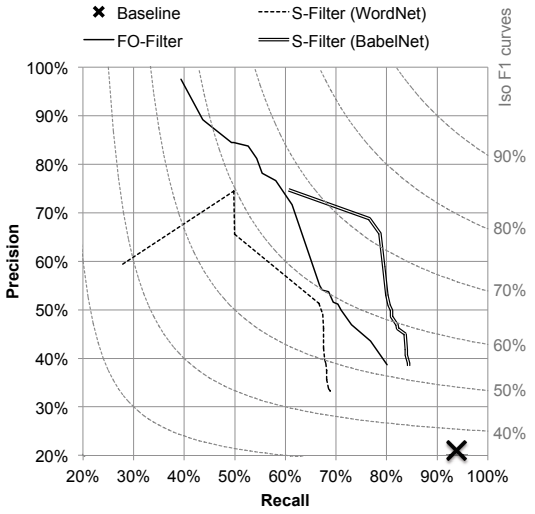


Fig. 3. RE performance of Web-DARE rules with different applied filters. Dashed curves in gray depict points with equal F1 score. For the semantic filter ("S-Filter"), the curves resulted from varying k from 1 to 15. FO-Filter is described in [22]. Results are averaged over seven relations.

Table 2. Impact of WordNet (WN) vs. BabelNet (BN) utilization on Web-DARE rule filtering. Results are averaged over seven relations, all values are in %.

k (Alg. 1)	Precision		Recall		F1	
	WN	BN	WN	BN	WN	BN
(Basel.)	21.00		93.83		34.32	
15	33.24	38.50	68.87	84.37	44.84	52.87
10	38.89	46.16	68.01	82.20	49.48	59.12
5	49.07	52.99	67.40	80.04	56.79	63.76
3	65.57	65.76	49.93	78.69	56.69	71.64
2	74.43	68.79	49.84	76.61	59.70	72.49
1	59.43	74.66	27.84	60.73	37.92	66.98

In contrast, the semantic filter trained with BabelNet does not permit precision levels above 75% for the average of the relations targeted in this paper, but it has at the same time a more reasonable precision-recall trade-off, e.g., by retaining about 15 percentage points recall above the FO-Filter at a precision level of around 70%. In the recall range covered by the BabelNet filter, its precision is consistently higher.

As illustrated by the chart, training the S-Filter with WordNet instead of BabelNet leads to inferior performance. Table 2 shows the Web-DARE RE performance for different parameter values of Algorithm 1. The use of BabelNet consistently leads to a higher F-score compared to WordNet. For example at $k = 2$, the F-score is roughly thirteen percentage points higher.

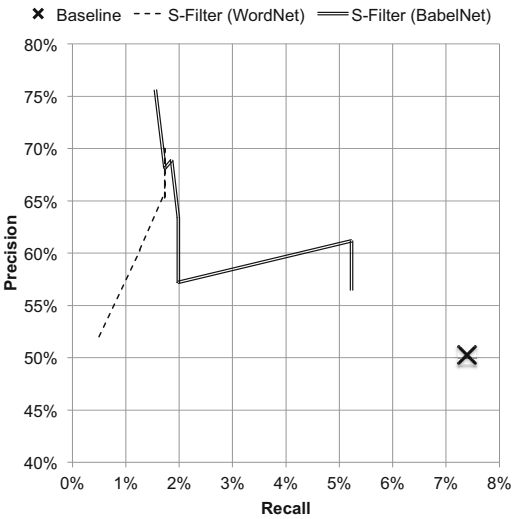


Fig. 4. RE performance of NELL rules, both with and without semantic filter ("S-Filter"). *k* varies from 1 to 15. Results are averaged over seven relations.

Semantic Filter for NELL Rules. Figure 4 shows the precision versus relative recall results of the baseline and our semantic filtering algorithms when applied to NELL's patterns. Again, the RE precision increases. The relative recall values on our test data do not permit any conclusions to be drawn for the NELL system. Due to the low number of mentions found in the NELL recall baseline (see Table 1), the filter application has a high impact on the depicted recall values and thus the curves show a non-monotonic growth. Nevertheless, as the chart indicates, the proposed filter can also be applied to pattern sets of different RE rule formalisms. Similarly to Figure 3, Figure 4 demonstrates that training the filter on BabelNet leads to superior RE performance compared to the filter variant trained on WordNet.

6.4 Result Analysis and Insights

Generality. Both Figures 3 & 4, as well as Table 2, show significant performance improvements after the application of the semantic filter, regardless of the underlying pattern formalism, i.e., dependency-analysis-based or surface-level-based. This means that our algorithm could be applied in a large variety of application scenarios, as long as the patterns or rules contain content words to which the semantic filter can be applied.

BabelNet vs. WordNet. The semantically-enhanced RE performance values of Web-DARE and NELL as given in Sections 6.2 & 6.3 fully support our initial expectation that BabelNet, with its richer inventory of lexical semantic relations, is better suited for effective rule filtering.

Consider the following example from the marriage relation:

(3) person $\xrightarrow{appos}$ widow $\xrightarrow{prep}$ of $\xrightarrow{pobj}$ person

This rule draws on the concept of deceased spouses, i.e. *widow*, for detecting the target relation. Since the semantic graph created with BabelNet contains this concept, the rule is identified as being useful for RE and hence it is not filtered out, in contrast to the filter from WordNet, which erroneously excludes it.

Individual Relations. The performance of the filter varies across relations. Due to space limitations we cannot show detailed per-relation results here. The filter works particularly well for relations like *acquisition* and *person birth/death*, whereas the results are rather discouraging for *place lived*. Investigating the sampled mentions of the latter relation, we found that this can be attributed to the larger lexical diversity of this relation. Often the semantic information is carried by constructions such as "Belfast writer J. Adams", where the lexical anchor "writer" is semantically insignificant to the relation. To get high coverage on such mentions extraction rules would have to match a certain set of semantically diverse nouns here, without matching all nouns ("Belfast visitor Cameron"). The relation seems to require much background knowledge, which may have to include entailment and other inferences. For example, a mention of a person being a senator for some (US) state could, depending on legal requirements, indeed be a mention for *place lived*.

Semantic Filter vs. FO-Filter. Finally, we investigated the causes of the superior performance of our new semantic filter compared to the pre-existing FO-Filter. In addition to the problem of always finding mutually exclusive relations with compatible entity signatures, the FO-Filter also has the disadvantage of not excluding erroneous rules which belong neither to the particular target relation nor to any of the compatible relations. In contrast, the new semantic filter works independently for each relation.

The following low-precision Web-DARE rules illustrate this point, all learned for the *marriage* relation:

(4) person $\xleftarrow{nsubj}$ lose $\xrightarrow{prep}$ to $\xrightarrow{pobj}$ person

(5) person $\xleftarrow{nsubj}$ date $\xrightarrow{dobj}$ person

(6) person $\xleftarrow{nsubj}$ meet $\xrightarrow{dobj}$ person

These rules, as they express typical relations for married couples, get strong statistical support for the *marriage* relation against the other relations. Therefore, the FO-Filter is not able to correctly identify them as wrong. In contrast, the semantic filter correctly disposes of them.

Another shortcoming of the FO-Filter is the recurring exclusion of high-quality patterns for which there is only limited support in the training data. When taking only the frequency of a pattern into account, these patterns cannot be distinguished from erroneously learned ones. Our use of an additional lexical-semantic resource, such as WordNet/BabelNet, provides a filtering mechanism that correctly identifies the appropriate meaning of the target relation. Consider the following example rule, which, as it has

a low frequency, gets filtered out by the FO-Filter, whereas, as it expresses a relevant word sense for the considered relation, gets classified as correct by our semantic filter:

(7) person $\xleftarrow{poss}$ widower $\xrightarrow{appos}$ person

7 Conclusion and Outlook

After the successful utilization of parsing for large-scale RE, the time seems ripe for injecting more semantics into this most challenging task within IE. This paper demonstrates that exploiting advanced comprehensive semantic knowledge resources can significantly improve extraction performance.

This is just the beginning, opening the way for new lines of research. The semantic classifier should now be extended for rule classification with respect to relations, building a bridge between traditional IE and open IE. The synonyms provided by semantic resources could also be applied to extend the rule set for increased coverage, in addition to filtering it. As a side result of the comparison between the FO-Filter and the new semantic filter, we observed that the two methods exhibit different shortcomings, giving rise to the hope that a combination may further improve RE performance.

Acknowledgements. This research was partially supported by the European Research Council through the "MultiJEDI" Starting Grant No. 259234, by the German Federal Ministry of Education and Research (BMBF) through the projects Deependance (contract 01IW11003) and Software Campus (contract 01IS12050, sub-project Intellektix) and by Google through a Faculty Research Award granted in July 2012.

References

1. Agichtein, E.: Confidence estimation methods for partially supervised information extraction. In: Proc. of the Sixth SIAM International Conference on Data Mining (2006)
2. Ballesteros, M., Nivre, J.: Maltoptimizer: An optimization tool for maltparser. In: Proc. of EACL, pp. 58–62 (2012)
3. Banko, M., Etzioni, O.: The Tradeoffs Between Open and Traditional Relation Extraction. In: Proc. of ACL/HLT, pp. 28–36 (2008)
4. Banko, M., Cafarella, M.J., Soderland, S., Broadhead, M., Etzioni, O.: Open information extraction from the Web. In: Proc. of the 20th IJCAI, pp. 2670–2676 (2007)
5. Betteridge, J., Carlson, A., Hong, S.A., Hruschka Jr., E.R., Law, E.L.M., Mitchell, T.M., Wang, S.H.: Toward never ending language learning. In: Proc. of the 2009 AAAI Spring Symposium on Learning by Reading and Learning to Read (2009)
6. Bollacker, K.D., Evans, C., Paritosh, P., Sturge, T., Taylor, J.: Freebase: a collaboratively created graph database for structuring human knowledge. In: Proc. of SIGMOD, pp. 1247–1250 (2008)
7. Brin, S.: Extracting patterns and relations from the World Wide Web. In: Proc. of WebDB, pp. 172–183 (1998)
8. Carlson, A., Betteridge, J., Kisiel, B., Settles, B., Hruschka Jr., E., Mitchell, T.: Toward an Architecture for Never-Ending Language Learning. In: Proc. of AAAI, pp. 1306–1313 (2010)
9. Carlson, A., Betteridge, J., Hruschka Jr., E.R., Mitchell, T.M.: Coupling semi-supervised learning of categories and relations. In: Proc. of the NAACL HLT 2009 Workskop on Semi-supervised Learning for Natural Language Processing (2009)

10. Carlson, A., Betteridge, J., Wang, R.C., Hruschka Jr., E.R., Mitchell, T.M.: Coupled semi-supervised learning for information extraction. In: Proc. of WSDM (2010)
11. Chan, Y.S., Roth, D.: Exploiting Syntactico-Semantic Structures for Relation Extraction. In: Proc. of ACL, pp. 551–560 (2011)
12. Chiarcos, C., Nordhoff, S., Hellmann, S.: Linked Data in Linguistics. Representing and Connecting Language Data and Language Metadata. Springer, Heidelberg (2012)
13. Etzioni, O., Fader, A., Christensen, J., Soderland, S.: Mausam: Open Information Extraction: The Second Generation. In: Proc. of IJCAI, pp. 3–10 (2011)
14. Fader, A., Soderland, S., Etzioni, O.: Identifying Relations for Open Information Extraction. In: Proc. of EMNLP, pp. 1535–1545 (2011)
15. Fellbaum, C.: WordNet: an electronic lexical database, Cambridge, MA, USA (1998)
16. Finkel, J.R., Grenager, T., Manning, C.: Incorporating non-local information into information extraction systems by gibbs sampling. In: Proc. of ACL, pp. 363–370 (2005)
17. Grishman, R., Sundheim, B.: Message understanding conference - 6: A brief history. In: Proc. of the 16th International Conference on Computational Linguistics, Copenhagen (June 1996)
18. Hoffart, J., Suchanek, F.M., Berberich, K., Weikum, G.: YAGO2: A spatially and temporally enhanced knowledge base from Wikipedia. Artificial Intelligence 194, 28–61 (2013)
19. Jiang, J., Zhai, C.: A Systematic Exploration of the Feature Space for Relation Extraction. In: Proc. of NAACL, pp. 113–120 (2007)
20. Kambhatla, N.: Combining lexical, syntactic, and semantic features with maximum entropy models for information extraction. In: Proc. of ACL (Demonstration), pp. 178–181 (2004)
21. Kozareva, Z., Hovy, E.H.: A semi-supervised method to learn and construct taxonomies using the Web. In: Proc. of EMNLP, pp. 1110–1118 (2010)
22. Krause, S., Li, H., Uszkoreit, H., Xu, F.: Large-scale learning of relation-extraction rules with distant supervision from the web. In: Cudré-Mauroux, P., Heflin, J., Sirin, E., Tudorache, T., Euzenat, J., Hauswirth, M., Parreira, J.X., Hendler, J., Schreiber, G., Bernstein, A., Blomqvist, E. (eds.) ISWC 2012, Part I. LNCS, vol. 7649, pp. 263–278. Springer, Heidelberg (2012)
23. Lao, N., Mitchell, T., Cohen, W.W.: Random walk inference and learning in a large scale knowledge base. In: Proc. of EMNLP, pp. 529–539 (2011)
24. Miller, S., Fox, H., Ramshaw, L., Weischedel, R.: A Novel Use of Statistical Parsing to Extract Information from Text. In: Proc. of NAACL, pp. 226–233 (2000)
25. Mintz, M., Bills, S., Snow, R., Jurafsky, D.: Distant supervision for relation extraction without labeled data. In: Proc. of ACL/AFNLP, pp. 1003–1011 (2009)
26. Mohamed, T., Hruschka, E., Mitchell, T.: Discovering relations between noun categories. In: Proc. of EMNLP, pp. 1447–1455 (2011)
27. Moro, A., Navigli, R.: WiSeNet: building a wikipedia-based semantic network with ontologized relations. In: Proc. of CIKM, pp. 1672–1676 (2012)
28. Moro, A., Navigli, R.: Integrating Syntactic and Semantic Analysis into the Open Information Extraction Paradigm. In: Proc. of IJCAI, pp. 2148–2154 (2013)
29. Nastase, V., Strube, M.: Transforming Wikipedia into a large scale multilingual concept network. Artificial Intelligence 194, 62–85 (2013)
30. Navigli, R.: Word Sense Disambiguation: A survey. ACM Comput. Surv. 41(2), 1–69 (2009)
31. Navigli, R., Ponzetto, S.P.: BabelNet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. Artificial Intelligence 193, 217–250 (2012)
32. Navigli, R.: BabelNet goes to the (Multilingual) Semantic Web. In: Proc. of MSW (2012)
33. Navigli, R., Ponzetto, S.P.: Joining forces pays off: Multilingual Joint Word Sense Disambiguation. In: Proc. of EMNLP-CoNLL, pp. 1399–1410 (2012)
34. Navigli, R., Ponzetto, S.P.: Multilingual WSD with Just a Few Lines of Code: the BabelNet API. In: Proc. of ACL (System Demonstrations), pp. 67–72 (2012)

35. Nguyen, Q., Tikk, D., Leser, U.: Simple tricks for improving pattern-based information extraction from the biomedical literature. Journal of Biomedical Semantics 1(1) (2010)
36. Nguyen, T.V.T., Moschitti, A.: Joint distant and direct supervision for relation extraction. In: Proc. of 5th IJCNLP, pp. 732–740 (2011)
37. Parker, R.: English Gigaword, 5th edn. Linguistic Data Consortium. Philadelphia (2011)
38. Pasca, M., Lin, D., Bigham, J., Lifchits, A., Jain, A.: Names and Similarities on the Web: Fact Extraction in the Fast Lane. In: Proc. of ACL/COLING (2006)
39. Ravichandran, D., Hovy, E.H.: Learning surface text patterns for a Question Answering System. In: Proc. of ACL, pp. 41–47 (2002)
40. Shinyama, Y., Sekine, S.: Preemptive Information Extraction using Unrestricted Relation Discovery. In: Proc. of HLT-NAACL (2006)
41. Soderland, S., Roof, B., Qin, B., Xu, S., Mausam, E.O.: Adapting Open Information Extraction to Domain-Specific Relations. AI Magazine 31(3), 93–102 (2010)
42. Suchanek, F.M., Kasneci, G., Weikum, G.: YAGO: A large ontology from Wikipedia and WordNet. J. Web. Semant. 6, 203–217 (2008)
43. Surdeanu, M., Ciaramita, M.: Robust information extraction with perceptrons. In: Proc. of the NIST 2007 Automatic Content Extraction Workshop, ACE 2007 (March 2007)
44. Surdeanu, M., Gupta, S., Bauer, J., McClosky, D., Chang, A.X., Spitkovsky, V.I., Manning, C.D.: Stanford's distantly-supervised slot-filling system. In: Proc. of TAC (2011)
45. Uszkoreit, H.: Learning relation extraction grammars with minimal human intervention: Strategy, results, insights and plans. In: Gelbukh, A. (ed.) CICLing 2011, Part II. LNCS, vol. 6609, pp. 106–126. Springer, Heidelberg (2011)
46. Volokh, A., Neumann, G.: Comparing the benefit of different dependency parsers for textual entailment using syntactic constraints only. In: Proc. of SemEval, pp. 308–312 (2010)
47. Weld, D.S., Hoffmann, R., Wu, F.: Using Wikipedia to bootstrap open information extraction. SIGMOD Record 37, 62–68 (2008)
48. Wu, F., Weld, D.S.: Open Information Extraction Using Wikipedia. In: Proc. of ACL (2010)
49. Wu, F., Hoffmann, R., Weld, D.S.: Information extraction from Wikipedia: moving down the long tail. In: Proc. of KDD, pp. 731–739 (2008)
50. Xu, F.: Bootstrapping Relation Extraction from Semantic Seeds. PhD thesis, Saarland University (2007)
51. Xu, F., Uszkoreit, H., Krause, S., Li, H.: Boosting relation extraction with limited closed-world knowledge. In: Proc. of COLING (Posters), pp. 1354–1362 (2010)
52. Xu, F., Uszkoreit, H., Li, H.: A seed-driven bottom-up machine learning framework for extracting relations of various complexity. In: Proc. of ACL (2007)
53. Xu, W., Grishman, R., Zhao, L.: Passage retrieval for information extraction using distant supervision. In: Proc. of IJCNLP, pp. 1046–1054 (2011)
54. Yangarber, R.: Counter-training in discovery of semantic patterns. In: Proc. of ACL (2003)
55. Yangarber, R., Grishman, R., Tapanainen, P.: Automatic acquisition of domain knowledge for information extraction. In: Proc. of COLING, pp. 940–946 (2000)
56. Yates, A., Cafarella, M., Banko, M., Etzioni, O., Broadhead, M., Soderland, S.: TextRunner: open information extraction on the Web. In: Proc. of HLT-NAACL (Demo), pp. 25–26 (2007)
57. Yates, A., Etzioni, O.: Unsupervised Resolution of Objects and Relations on the Web. In: Proc. of HLT-NAACL, pp. 121–130 (2007)
58. Zelenko, D., Aone, C., Richardella, A.: Kernel methods for relation extraction. The Journal of Machine Learning Research 3, 1083–1106 (2003)
59. Zhou, G., Qian, L., Fan, J.: Tree kernel-based semantic relation extraction with rich syntactic and semantic information. Inf. Sci. 180(8), 1313–1325 (2010)
60. Zhou, G., Zhang, M.: Extracting relation information from text documents by exploring various types of knowledge. Inf. Process. Manage. 43(4), 969–982 (2007)

Semantic Message Passing for Generating Linked Data from Tables*

Varish Mulwad, Tim Finin, and Anupam Joshi

University of Maryland, Baltimore County
Baltimore, MD 21250 USA
{varish1,finin,joshi}@cs.umbc.edu

Abstract. We describe work on automatically inferring the intended meaning of tables and representing it as RDF linked data, making it available for improving search, interoperability and integration. We present implementation details of a joint inference module that uses knowledge from the linked open data (LOD) cloud to jointly infer the semantics of column headers, table cell values (e.g., strings and numbers) and relations between columns. We also implement a novel *Semantic Message Passing* algorithm which uses LOD knowledge to improve existing message passing schemes. We evaluate our implemented techniques on tables from the Web and Wikipedia.

Keywords: Tables, Semantic Web, Linked Data, Graphical Models.

1 Introduction

Tables are an integral part of documents, reports and Web pages, compactly encoding important information that can be difficult to express in text. Table-like structures outside documents, such as spreadsheets, CSV files, log files and databases, are widely used to represent and share information. A Google study [2] found more than 150 million high quality relational tables on the Web. Many governments share public data useful to citizens and businesses as tables. The U.S. government's data sharing website, for example, had nearly 400,000 such datasets as of September 2012. Medical researchers can asses treatment efficacy via a meta-analysis of previously published clinical trials, often using systems like MEDLINE[1] to find relevant articles and extract key data, which is typically summarized in tables, like the one in Figure 1.

Integrating and searching over this information benefits from a better understanding of its intended meaning, a task with several unique challenges. The very structure of tables which adds value and makes it easier for human understanding also makes it harder for machine understanding. Web search engines, for example, perform well when searching over narrative text on the Web, but poorly when searching for information embedded in tables in HTML documents.

* An extended version of this paper with additional material is available at http://ebiquity.umbc.edu/p/627

[1] http://nlm.nih.gov/bsd/pmresources.html

H. Alani et al. (Eds.): ISWC 2013, Part I, LNCS 8218, pp. 363–378, 2013.

We might interpret tables using proven NLP techniques; after all, tables also contain text. We understand the meaning of a sentence by understanding the meaning of the individual words, which in turn are understood using grammatical knowledge and the context provided by the surrounding text. Contrast that with a table like Figure 1, where uncovering its meaning requires interpreting the row and column headers, the relation between them and mapping cell values to appropriate measurements.

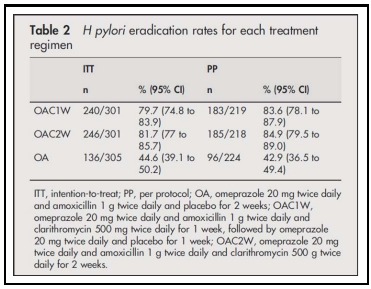


Table 2 *H pylori* eradication rates for each treatment regimen

	ITT		PP	
	n	% (95% CI)	n	% (95% CI)
OAC1W	240/301	79.7 (74.8 to 83.9)	183/219	83.6 (78.1 to 87.9)
OAC2W	246/301	81.7 (77 to 85.7)	185/218	84.9 (79.5 to 89.0)
OA	136/305	44.6 (39.1 to 50.2)	96/224	42.9 (36.5 to 49.4)

ITT, intention-to-treat; PP, per protocol; OA, omeprazole 20 mg twice daily and amoxicillin 1 g twice daily and placebo for 2 weeks; OAC1W, omeprazole 20 mg twice daily and amoxicillin 1 g twice daily and clarithromycin 500 mg twice daily for 1 week, followed by omeprazole 20 mg twice daily and placebo for 1 week; OAC2W, omeprazole 20 mg twice daily and amoxicillin 1 g twice daily and clarithromycin 500 g twice daily for 2 weeks.

Fig. 1. Tables in clinical trials reports [22] often have both row and column headers, contain numerical data and have captions with critical metadata

The intended meaning of tables is strongly suggested by the column and row headers, cell values and relations between the columns. Additional context can often be found in a caption or other text near the table. How does one capture this intended meaning? Consider the leftmost column in the table shown in Figure 2. The column header *City* represents the class and the values *Baltimore*, *Philadelphia*, *New York* and *Boston* are instances of that class. Capturing the relationships between table columns can help confirm or deny prior understanding. Consider the strings in the third column of the table in Figure 2. An initial analysis of the column might suggest that they refer to *Politicians*. Additional information that strings in column one represent cities, can help confirm that they are not only *Politicians* but also *Mayors* of the cites mentioned in the first column.

City	*State*	*Mayor*	*Population*
Baltimore	MD	S.Rawlings-Blake	640,000
Philadelphia	PA	M.Nutter	1,500,000
New York	NY	M.Bloomberg	8,400,000
Boston	MA	T.Menino	610,000

Fig. 2. A table with information about U.S. cities

Our goal is to encode a table as RDF linked data, mapping columns to appropriate classes, linking cell values to entities, literal constants or implied measurements, identifying relations between columns and asserting appropriate RDF triples. We describe an extensible, domain-independent framework to do this using background knowledge from an LOD resource. A novel feature is the incorporation of semantic knowledge in the message passing algorithm used for joint assignments in a graphical model.

Producing linked data representation is a complex task that requires developing an overall understanding of the intended meaning of the table as well as choosing the right URIs to represent its schema and instances. We decompose it as follows: (1) assign column (and/or row) headers classes from an appropriate ontology, (2) link cell values to literals or entities (creating them as necessary), (3) discover relations between table columns and add properties to represent them, and (4) generate a linked data representation. We describe our approach to these tasks and an evaluation of the results in the remainder of the paper.

2 Approach

Figure 3 shows our extensible and domain independent framework for inferring the meaning of a table and representing it explicitly as linked data. An input table first goes through a *preprocessing phase* with modules to handle a number of pragmatic issues, such as sampling rows from large tables and recognizing and expanding acronyms and stylized literal values (e.g., phone numbers). These modules can be developed independently and added to the framework without affecting others or hampering the workflow. Puranik [14], for example, developed modules to identify whether a column in a table consists of commonly encoded data such as *SSN*, *zip codes*, *phone numbers* and *addresses*.

A table is then processed by the *query and rank* module which queries the background LOD sources to generate initial ranked lists of candidate assignments for column headers, cell values and relations between columns. Once candidate assignments are generated, the *joint inference* component uses a probabilistic graphical model to capture the correlation between column headers, cell values and column relations to make class, entity and relation assignments. After the mapping is complete, linked data triples are produced. Although our goal is to develop a fully automated system achieving a high level of accuracy, we recognize that practical systems will benefit from or even require human input. Future work is planned to allow users to view the interpretation and give feedback and advice if it is incorrect.

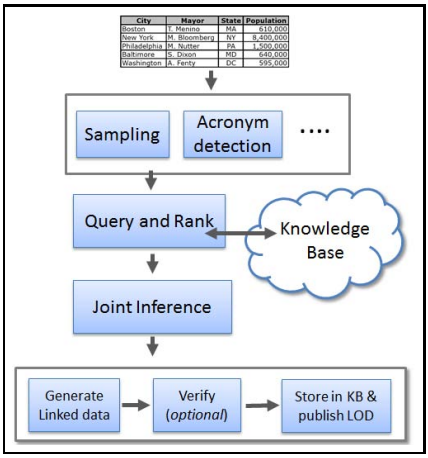


Fig. 3. Our extensible and domain independent framework relies on a *joint inference* module to generate a representation of the meaning of the table as a whole

While we presented a brief sketch of our framework previously [11], the contributions of this paper include (1) an implementation of the graphical model with improved factor nodes, (2) enhancements to our novel *Semantic Message Passing* algorithm and a detailed description of its implementation, and (3) a thorough evaluation. The rest of the section consists of a brief review of the *query and rank* module followed by details of the *joint inference* module, which is the key focus of this paper.

2.1 Query and Rank

The *query and rank* module generates an initial ranked list of candidate assignments for the column headers, cell values and column relations using data from DBpedia [1], Yago [17] and Wikitology [18]. For most of the general tables, especially ones found on the Web, these knowledge sources provide good coverage.

Additional LOD data sources can be selected and incorporated, automatically or manually, based on the table's domain.

Generating and Ranking Candidates for Cell Values. We generate an initial set of candidate entities for each cell value using Wikitology, a hybrid knowledge base combining unstructured and structured information from Wikipedia, DBpedia and Yago. The contents of the column header and other row values are used as context when querying Wikitology. The query for *Baltimore*, for example, consists of the query string *Baltimore* and the context data *City*, *MD*, *S.C.Rawlings-Blake*, and *640,000* [12]. Wikitology returns ranked lists of entities and classes, which for *Baltimore*, include the entities *Baltimore*, *John_Baltimore* and *Baltimore_Ravens* along with DBpedia classes *City*, *PopulatedPlace* and *Place* and Yago types *CitiesInMaryland* and *GeoclassPopulatedPlace*. An *entity ranker* then re-ranks a cell's candidates entities using an approach adapted from [4] and features from [12] to return a measure of how likely the given entity (e.g., *John_Baltimore*) is the correct assignment for the string mention (e.g., *Baltimore*).

Generating Candidates for Columns. Initial candidate classes for a column are generated from its cell values, each of which has a set of candidate entities, which in turn have sets of DBpedia and Yago classes. The column's potential classes is just the union of the classes from the its cells. We generate two separate set of candidate classes – one for DBpedia classes and another for Yago classes.

Generating Candidate Relations between Columns. Identifying relations between table columns is an important part of table understanding and is modeled by finding appropriate predicates from the reference LOD's ontologies (e.g., DBpedia). We generate candidate relations for every pair of columns in the table, based on the cell value pairs in the respective columns. Each cell value has a set of candidate entities, which in turn may be linked to other entities in the reference LOD resources. For example, the DBpedia entities *Baltimore* and *Maryland* are linked via the predicates *isPartOf* and *subdivisionName*.

We use the links between pairs of entities to generate candidate relations. For a pair of cell values in the same row between the two columns, the candidate entity sets for both cells are obtained. For each possible pairing between the entities in both the candidate sets, we query Yago and DBpedia, to obtain relation in either direction i.e. *entityrow1 someproperty1 entityrow2* and *entityrow2 someproperty2 entityrow1*. This gives us a candidate set between pair of row cell values. The candidate relation set for the entire column pair is generated by taking a union of the set of candidate relations between individual pairs of row cell values. Thus for example, the candidate relations between column *City* and column *State* might include *isPartOf, capitalCity, bornIn* etc. Again, we generate two sets of candidate relations, one from DBpedia and the other from Yago.

Literal Constants. We use a regular expression to distinguish string mentions, which probably refer to entities, and literal constants such as numbers and measurements, which probably do not. If the cell value is a literal constant, candidate entities are not generated and the cell is mapped to NO-ANNOTATION.

If all the cells in a column are literals, we update the column header annotation to NO-ANNOTATION.

2.2 Joint Inference

Once the initial sets of candidate assignments are generated, the joint inference module assigns values to columns and row cell values and identifies relations between the table columns. The result is a representation of the meaning of the table as a whole. Probabilistic graphical models [8] provide a powerful and convenient framework for expressing a joint probability over a set of variables and performing inference or joint assignment of values to the variables. Probabilistic graphical models use graph based representations to encode probability distribution over a set of variables for a given system. The nodes in such a graph represent the variables of the system and the edges represent the probabilistic interaction between the variables.

We represent a table as a Markov network graph in which the column headers and cell values represent the variable nodes and the edges between them represent their interactions. The edges in a Markov network graph are undirected because the interactions between the variables are symmetrical. In the case of tables, interactions between the column headers, table cell values and the relation between table columns are symmetrical and thus a Markov network is well suited for tables.

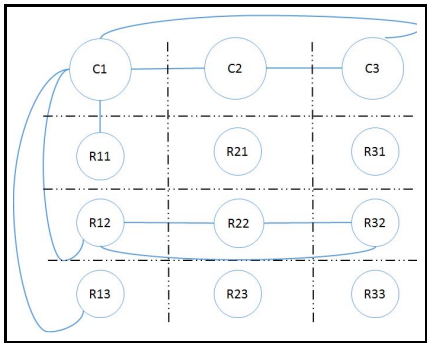


Fig. 4. This graph represents the interactions between the variables in a simple table. Only some of the connections are shown to keep the figure simple.

Figure 4 shows interaction between the column headers (represented by C_i where $i \in 1$ to 3) and cell values (represented by R_{ij} where $i, j \in 1$ to 3). In a typical well-formed table, each column contains data of a single syntactic type (e.g., strings) that represent entities or values of a common semantic type (e.g., people). For example, in a column of cities, the column header *City* represents the semantic type of values in the column and *Baltimore*, *Boston* and *Philadelphia* are instances of that type. Thus, knowing the type of the column header, influences the decision of the assignment to the table cells in that column and vice-versa. To capture this interaction, we insert an edge between the column header variable and each of the cell values in that column.

Table cells across a given row are also related. Consider a table cell with a value *Beetle*, which might refer to an *insect* or a *car*. Suppose an adjacent cell has a string value *red*, which is a reference to a *color*, and another cell in the same row has the string value *Gasoline*, which is a type of *fuel source*. As a collection, the cell values suggest that the row represents values of a *car* rather than an *insect*. Thus, the interpretation of each cell is influenced by the interpretation of the others in its row. This co-relation when considered between pairs of table

cell values between two columns can also be used to identify relations between table columns. To capture this context, we insert edges between all the table cells in a given row.

Similar interactions exist between the column headers. By itself, the column header *City* suggests that column's cells might refer to city instances. However, if the other columns appear to refer to basketball players, coaches and basketball divisions, we can infer that the cities column refers to a team itself. This is an example of metonymy, in which an entity (i.e., the team) is referenced by one of its significant properties (i.e., the location of its base). This interaction is captured by inserting edges between column header variables.

To perform any meaningful inference over the graph, it must be parameterized. We do so by representing the graph in Figure 4 as a factor graph as shown in Figure 5. The graph's square nodes represent what are known as '*factor nodes*', which compute and capture affinity or agreement between interacting variables. For example, ψ_3 in Figure 5 computes the agreement between the class assigned to column header and entities linked to the cell values in that column; ψ_4 between row cell values for a given pair of columns and ψ_5 between column headers.

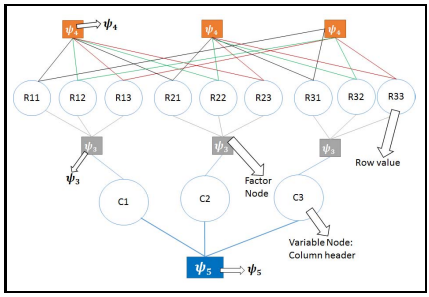


Fig. 5. This factor graph is a parameterized Markov network wherein the square nodes represent factor nodes

Semantic Message Passing. Factor nodes allow the joint inference process to operate. Typical inference algorithms such as *belief propagation* and *message passing* rely on pre-computed joint probability distribution tables (PDTs) stored at the factor nodes. For example, the factor node ψ_3 for column header variable C_1 would store a PDT over the variables $C_1, R_{11}, R_{12}, R_{13}$; i.e. ψ_3 would pre–compute and store a PDT over the column header and all row cell values. As the size of the candidate set of values that C_i and R_{ij} can be mapped to increases, the size of the PDT will rapidly grow. Assuming that the size of the candidate set for a variable is 25, ψ_3 associated with the variables $C_1, R_{11}, R_{12}, R_{13}$ would have *390,625* entries in the joint PDT!

We implement a variation of an inference algorithm which incorporates semantics and background knowledge from LOD to avoid the problem of computing large joint PDTs at factor nodes. Our *Semantic Message Passing* algorithm is conceptually similar to the idea of Message Passing schemes.

The variable nodes in the graph send their current assignment to the factor nodes to which they are connected. For example, R_{11} sends its current assignment to factor nodes ψ_3 and ψ_4. Once the factor nodes receive values from all connected variable nodes, they compute agreement between the values. Thus, in one of the iterations, ψ_3 might receive values *City*, *Baltimore_Ravens*,

Philadelphia, *New_York* and *Boston*. The goal of ψ_3 is to determine if all the assignments agree and, if not, identify the outliers.

In this case ψ_3 identifies *Baltimore_Ravens* as an outlier and sends a CHANGE message to R_{11}, together with its semantic preferences for a new, alternate value that R_{11} might produce. In our example, ψ_3 informs R_{11} of its preference for update to an entity of type *City*. To the rest of the variable nodes, ψ_3 sends a NO-CHANGE message. This process is performed by all factor nodes. Once a variable node receives messages from all of its connected factor nodes, it decides whether to update its value or not.

If it receives a message of NO-CHANGE from all factor nodes, its current assignment is in agreement with the others and it need not update its assignment. If it receives a CHANGE message from some or all factor nodes, it updates its current assignment, taking into consideration the semantic preferences provided by the factor nodes. The entire process repeats until convergence, i.e., agreement over the entire graph is achieved. A hard convergence metric could be to repeat the process until no variable node receives a CHANGE message.

Our *Semantic Message Passing* algorithm thus circumvents the problem of computing joint PDTs at factor nodes by computing agreement over current assigned values. Furthermore, our scheme not only detects individual variable nodes that have incorrect assignments, but provides the nodes with guidance on the characteristics or *semantics* associated with the value that a variable node should update to. This capability requires defining semantically-aware factor nodes that can perform such functions. In this paper, we describe our implementation of factor nodes ψ_3 and ψ_4 and the process by which a variable node updates its values based on the messages received and our metric for graph convergence.

ψ_3 – Column Header and Row Cell Value Agreement Function. The ψ_3 factor node computes agreement between the class assigned to the column and the entities assigned to its cell values. For example, agreement between the column assigned type *City* and candidate cell assignments *Baltimore_Ravens*, *Philadelphia*, *New_York* and *Boston*. Recall that at the end of the *query and rank* phase, every row cell value has an initial entity assignment and every column header has a set of candidate classes. In our current implementation every column header C_i maintains two separate sets of candidate classes – one from Yago's classes and the other from DBpedia's. Each cell value in a column is mapped to an initial entity e which in itself has its own set of Yago and DBpedia classes. The initial entities assigned to a column's cell values perform a majority voting over the Yago and DBpedia class set to pick the top Yago and DBpedia class. Each entity votes and increments the score of a class from the candidate set by 1 if the class is present in the class set associated with e.

The Yago and DBpedia candidate class sets are ordered by votes. ψ_3 computes the top score for each of the top classes. The top score is simply equal to the number of votes for the top class divided by the number of rows in the column. Ideally, we want to pick more specific classes (e.g., City) over general classes (e.g., Place) when making an assignment to the column headers. Thus, if multiple Yago classes get

voted as top class, we use a 'granularity' score as tie-breaker. The 'granularity' score is computed by simply dividing the number of instances that belong to the class by the total number of instances and subtracting the result from one. This assigns a higher score to specific classes and a lower score to general classes.

Once the top class(es) are identified and their scores computed, ψ_3 determines if they can be used in the process of identifying cell values with incorrect assignments. It checks whether the top scores for the classes are below a certain threshold. If so, it implies lower confidence and agreement between row cell values and that the top classes cannot be relied upon. In such scenarios, ψ_3 sends a message of LOW-CONFIDENCE and NO-CHANGE to the variable nodes and also maps the column header class to NO-ANNOTATION.

If scores for both the top Yago and DBpedia classes are above the threshold, ψ_3 assigns both the classes to the column header and uses them in the process of identifying its cell values with incorrect assignments. However if either class is below threshold, it checks if the classes are aligned. We define the two classes as aligned if either the DBpedia class is a subclass of the Yago class or vice-versa. The subclass relation between the DBpedia and Yago classes is obtained via the PARIS project [16].

If the alignment exists, then ignoring the lower score to either Yago or DBpedia, ψ_3 picks both the classes as the Yago and DBpedia assignments for the column header respectively. Otherwise, the class with the lower score is ignored, and the other one is selected as the column class. Once the column header is mapped to a class assignment, ψ_3 revisits each entity assignment in the column. All cell values (variable nodes) whose currently assigned entity e include the top class(es) in their class set are sent a message of NO-CHANGE. Ones whose entity do not contain the top class in their class set are sent a CHANGE message. These variable nodes are also provided with the top class(es) as semantic preferences that their next entity assignment should try to fulfill. ψ_3 also sends the top score as a confidence score associated with the message.

ψ_4 – **Relations between Pair of Columns.** The goal of factor node ψ_4 is to discover if a relation exists between a pair of columns, say *City* and *State*, and, if so, to use it as evidence to uncover any incorrect entity assignments in the columns' cells. At the end of *query and rank* phase, every row cell value has an initial entity assignment, e. The pair of column headers is also associated with a set of candidate relations. The initial assigned pair of entities in the two columns perform majority voting to select the best possible relation from the candidate relation set. Each pair of entities in every row between the two columns votes for a relation rel. It increments its score by 1 if $< e_{i,k} > < rel > < e_{j,k} >$ or $< e_{j,k} > < rel > < e_{i,k} >$ is true (*here i, j refer to two columns and k refers to entities from the k th row between the two columns*). Factor node ψ_4 queries Yago and DBpedia separately to check if the relations exists. The current ψ_4 implementation also maintains two separate sets of candidate relations (one for Yago and the other for DBpedia) ordered by votes.

Factor node ψ_4 also computes a value for *topScore* for both top Yago and DBpedia relations as the number of votes divided by the number of cells in the

column. If a score is below the current threshold, the relations are discarded and ψ_4 sends LOW-CONFIDENCE and NO-CHANGE messages to all of the row cell values in both columns and updates the relation between the columns to NO-ANNOTATION. If the scores are above threshold, the top Yago and DBpedia relations between the two columns are updated. ψ_4 then revisits the pair of entities from the columns to discover possible incorrect assignments. For every currently assigned entity e in the two columns, ψ_4 checks if e appears as a subject or object of the top relations (depending upon the relation direction; either Yago or DBpedia). If e satisfies this constraint, a NO-CHANGE message is sent to the row cell and a CHANGE message is sent otherwise. The ψ_4 factor node also sends the relation information as characteristics the row cell should use for picking the next entity assignment and $topScore$ as confidence score associated with the message.

Updating Entity Annotations for Row Cell Value Variables. Every row cell r in the table receives messages from two types of factor nodes – column header factor node (ψ_3) and relation factor node (ψ_4). While r will receive only one message from the column header (since r belongs to only one column), it might receive multiple messages from relation factor nodes if its column is related to several columns in the table. For example, the column *City* is associated with columns *State*, *Mayor* and *Population*. The result is that r can receive conflicting messages – some factor nodes might send a CHANGE message, while others a NO-CHANGE message.

If all of the messages received by r are NO-CHANGE, r does not update. If all messages received by r are CHANGE, it decides to update its assignment. In the case of conflicting messages, r uses the confidence score sent by each factor node along with the message to compute the average score associated with the CHANGE messages and compares it against the average score associated with the NO-CHANGE messages. If the average CHANGE message score is higher, r updates its current assignment, otherwise it does not.

When r chooses to update its current assignment, it picks a new assignment based on the semantic preferences sent by the factor nodes. For example, a row cell value in the first column of the table in Figure 2 might receive messages to update to an entity which will have a *rdf:type City* and is the subject of relations *isPartOf* and *hasMayor*. The row cell value r iterates through its ranked list of candidate entity set and picks the next best entity satisfying all the semantic preferences specified in the message. In cases where r cannot find an entity that satisfies them all, it orders them based on the confidence scores associated with the respective messages and attempts to pick an entity assignment that satisfies the highest rank combination. For example, if there were three preferences, ranked *1*, *2* and *3*, r will first attempt to find an entity that satisfies*[1,2,3]* followed by *[1,2]* ; *[1,3]*; *[2,3]*; *[1]*; *[2]* and so on. If r is unable to find an entity that satisfies any preferences, then it updates its current assignment to NO-ANNOTATION.

An exception to this process occurs when the candidate entities for the cell all have low confidence (i.e., below the threshold ($index\_threshold$)). This is typically the case if the entity is absent from the knowledge base. In such cases, the algorithm maps the row cell value to NO-ANNOTATION rather than linking to any candidate.

If the column header is mapped to NO-ANNOTATION, the row cell values retain the top ranked entity assignment as suggested by the entity ranker.

Halting Condition. Once the row cell values have updated, they send their new assignments to the factor nodes and the entire process repeats. Ideally, the process should be repeated until best possible assignments are achieved; i.e., repeat until no variable node receives a CHANGE message or none of the variable nodes select a new assignment. Practically, this a hard convergence metric and it is often not achieved. In our current implementation, the *Semantic Message Passing* algorithm lets this cycle repeat for five iterations. After five iterations, the algorithm checks the number of variables that have received a CHANGE message. If the number of variables is lower than the threshold required to update the column header or relation annotation, the process is stopped, else the process continues until convergence or the tenth iteration has been completed. The assignments at the end of the final iteration are chosen as final values.

3 Evaluation

We begin by describing the experimental setup and follow by presenting our evaluation and analysis for column header, cell value and relation annotations, and performance of the graphical model in terms of convergence and running times.

Experimental Setup. We used tables from four different sets in our evaluation (see Figure 6). The original table sets, obtained from [10], include ground truth annotations in which column headers are mapped to Yago classes, relations between columns to Yago properties and row cell values linked to Yago entities. However, we could not use these assessments for column headers and relations, since our system uses data from both DBpedia and Yago, and thus developed our own gold standard. We ran our factor nodes ψ_3 and ψ_4 at low threshold (5%) to generate candidate classes and relation between table columns. We presented these candidates along with raw tables to human annotators, who marked each as *vital*, *okay* or *incorrect* (as in [20]). For example, annotators could mark the label *City* as *vital*, *PopulatedPlace* as *okay* and *Person* as incorrect for the first column of the table in Figure 2. Thus each column can have multiple *vital* and *okay* class labels as per annotator judgment.

Dataset	*Col & Rel*	*Cell Value*	Avg. [Col,Row]
Web_Manual	150	371	[2,36]
Web_Relation	28	–	[4,67]
Wiki_Manual	25	39	[4,35]
Wiki_Links	–	80	[3,16]

Fig. 6. Number of tables and the average number of columns (col) and rows in the sets used for column header, cell value and relation annotation. Average is over the number of tables used in cell value annotation.

For the evaluation that follows, we used the framework with the following values: for every cell value we chose the top 25 candidate entities from the entity ranker; the column header top score threshold used by ψ_3 and the relation top score threshold used by ψ_4 were set to 0.5; *index_threshold* used by a row cell value to determine low confidence entities was set to 10. The number of joint inference model iterations is as described in the previous section.

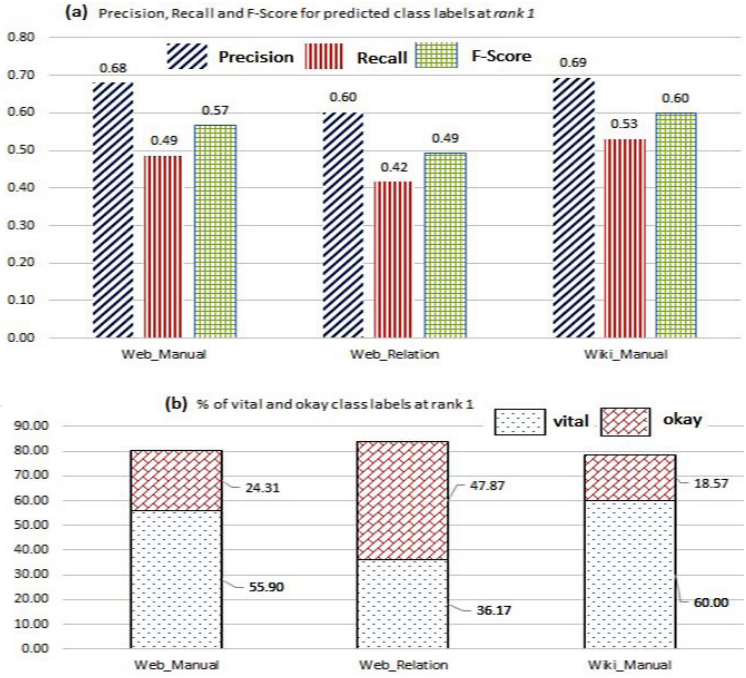


Fig. 7. (a) Precision, Recall and F-scores for column header annotations. (b) % of *vital* and *okay* class labels at rank 1.

Evaluating Column Header Annotations. We generated a ranked list of at most top ten class annotations for every column, with NO-ANNOTATION as the single value if no appropriate class was found. We compared the set of generated classes to those obtained from annotators, computing precision and recall for each k between 1 to 10. For precision, we assign a score of 1 for every *vital* class and 0.5 for every *okay* class identified by the framework. For recall, we assign 1 for every *vital* and *okay* class identified. This evaluation scheme is similar to the one described in [20].

Figure 7(a) shows the annotation results for class labels at rank 1 across three different sets. We observed that a single column had more than one label marked *vital* or *okay* by annotators and the recall increasing with k. The lower recall can be explained by the combination of having multiple labels for each column and only one label retrieved at *rank 1*. Due to space constraint, we do not show results for precision and recall for values of k between two and ten.

We also computed how often the label predicted at rank 1 was either *vital* or *okay*, as shown in Figure 7(b). The high percentage of a combination of *vital* and *okay* labels at rank 1 for all three sets (*79% or greater*) indicates that the top ranked labels are relevant. The results for the three sets are for the classes from the DBpedia ontology. We note that our F–scores for *Web_Manual (0.57)* and *Wiki_Manual (0.60)* are better than the previously reported scores of 0.43 and 0.56 in [10]. We fare slightly poorer for both datasets against scores of 0.65 and 0.67 as reported

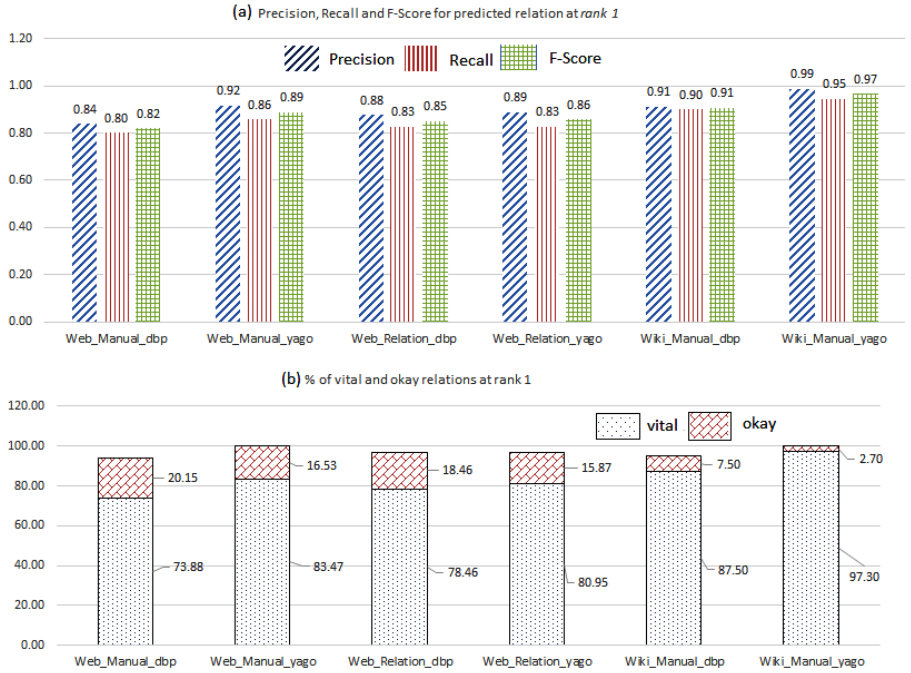


Fig. 8. (a) Precision, Recall and F-scores for relation annotations. (b) % of *vital* and *okay* relation labels at rank 1, as predicted by the framework. Dataset names followed by a *_dbp* are results for DBpedia relations; whereas the ones followed by *_yago* are for Yago relations.

in [20]. However, we note that their evaluation is over the annotations of the original ground truth in which every column header had only one, or sometimes two, correct classes. This leads to a better recall at rank 1, as compared to multiple correct classes, as in our case, with a combination of *vital* and *okay* labels. We also note that the task of the system in [20] is predicting classes for column headers and relation between "primary column" (e.g., a key) and other columns in the table independently. Our framework, in contrast, attempts to jointly map column header, cell values, relation between columns to appropriate assignments which in certain cases can lead to incorrect assignments.

Evaluating Relation Annotations. We generated and ranked the best ten relations (as RDF properties) both from DBpedia and Yago. We compared the set of relations generated to those obtained from the annotators, computing precision and recall for k from 1 to 10 as described in the previous section. Figure 8(a) shows the results across four different sets for both Yago and DBpedia relations. While it is plausible for a column header to have multiple *vital* and *okay* classes, the same may not hold true for relation between columns. We observed in our annotations that for every pair of columns, the set of *vital* and *okay* relations was smaller.

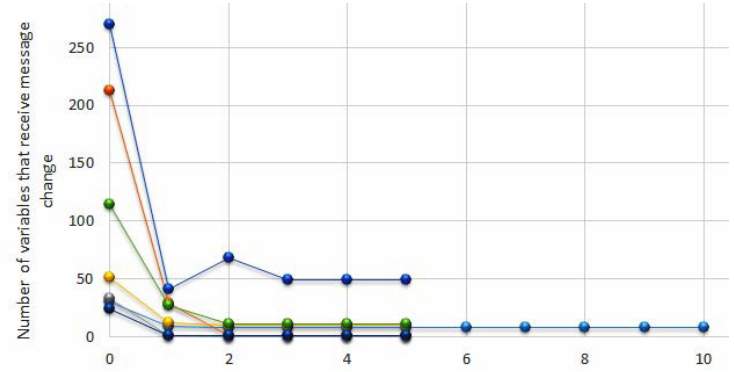


Fig. 9. X is the iteration number and Y the number of variables that received CHANGE message in that iteration. The value at x=0 is the number of cells in the table.

We explored possible relations between all pairs of columns, even though tables typically represent a small number of relations. Typically a table's primary (e.g., key) column participates in binary relations with many or most of the other columns in the table. Thus our system generates a fair number of NO-ANNOTATION labels. We believe both these reasons explain higher values for precision and recall. We also compute the percentage of *vital* and *okay* relations predicted by the framework at rank 1 (see Figure 8 (b)). Analogous to precision and recall results, a high percentage of labels were *vital* and *okay* indicating that our framework is generating relevant labels. Our F–scores for relation annotations for the datasets *Web_Manual_yago (0.89)*, *Web_Relation_yago(0.86)* and *Wiki_Manual_yago(0.97)* fare better as compared to the scores of 0.51, 0.63, 0.68 reported in [10].

Evaluating Cell Value Annotations. We compared the entity links generated by our framework to those obtained as ground truth from the original dataset [10]. Our framework linked the table cell value to an entity from DBpedia wherever possible, else it linked to NO-ANNOTATION. If our predicted entity link matched the ground truth, we considered it as a correct prediction, else incorrect. We obtained an accuracy of 75.89% over the *Wiki_Links* dataset; 67.42% over the *Wiki_Manual* dataset and 63.07% over the *Web_Manual* dataset. Lower accuracy for entity linking is likely due to the lack of relevant data in Yago and DBpedia. Although our framework might have discovered the correct assignments for column header and relation, if the entity did not have the class and relation information present on DBpedia or Yago, our framework will fail to find it.

For example, even if we discover the Yago class *YagoGeoEntity* which links all places, the DBpedia *Berlin* entity does not have that class, thus can lead to an incorrect assignment. Lower accuracy may also stem from the size of the candidate entity set. We restricted the size of the candidate entity set to 25; thus it is possible that the correct assignment could be outside this set. We also note certain discrepancy in the ground truth annotations. We discovered cases where

we were able to discover a correct entity annotation, whereas the ground truth said it was NO-ANNOTATION, which led to counting our annotation as incorrect. We are working on improving the ground truth annotation and we have not incorporated the results from the updated ground truth.

Graph Convergence. Figure 9 gives insight into how quickly the graph converges, showing the number of variable nodes that receive a CHANGE message at the end of every iteration (for eight tables). For most tables, the number of variable nodes that receive a CHANGE message stabilize after the first iteration. We had few cases, where the variable count fluctuated, i.e., increasing and decreasing as iterations increased. We also noticed cases where the variable count does not go to zero. Some number of "stubborn" variables keep receiving a CHANGE message at the end of every iteration, but cannot find a new value. However, we noticed that the number of stubborn variables are less as compared to the original number of variables in the table. We present results for eight tables for visual purposes; the results are representative of rest of the tables in the dataset. The average time required for the inference model across all tables was 3.4 seconds.

Entity Ranker. The *entity ranker* uses a classifier that produces likelihoods that strings should be linked to entities. The training and test datasets were generated using the ground truth for entity annotations from *Wiki_Links* set. For every string mention in the table, we queried Wikitology to get candidate entities and then computed feature values for the string similarity and popularity metrics for each mention/entity pair. A class label of 1 was assigned if the candidate entity was the correct assignment (available via ground truth in the dataset) and a 0 otherwise. The training set included 600 instances, evenly split between positive and negative instances. The test set included in all 681 instances with 331 positive and 350 negative instances. Out of the 681 instances, the model was able to correctly classify 619 instances with an accuracy of *90.9 %*. The precision, recall and F-score are presented in Figure 10.

Class	*Precision*	*Recall*	*F-Score*
0	0.959	0.849	0.901
1	0.871	0.966	0.916

Fig. 10. Precision, recall and F-score for the Naive Bayes model

4 Related Work

Our work is related to two threads of research, one focused on pragmatically generating RDF from databases, spreadsheets and CSV files and a more recent one that addresses inferring the implicit semantics of tables. Several systems have been implemented to generate semantic web data from databases [15,19,13], spreadsheets [6,9] and CSV files [3]. All are manual or only partially automated and none has focused on automatically generating *linked* RDF data for the entire table. In the domain of open government data, for example, [3] presents techniques to convert raw data (CSV, spreadsheets) to RDF but the results do not use existing classes or properties for column headers, nor does it link cell values to entities from the LOD cloud. Generating richer, enhanced mappings requires a manually constructed configuration file.

Early work in table understanding focused on extracting tables from documents and web pages [7,5] with more recent research attempting to understand their semantics. Wang et al. [21] began by identifying a single 'entity column' in a table and, based on its values and rest of the column headers, associate a concept from the Probase knowledge base with the table. Their work does not attempt to link the table cell values or identify relations between columns. Ventis et al. [20] associate multiple class labels (or concepts) with columns in a table and identify relations between the 'subject' column and the rest of the columns in the table. Their work also does not attempt to link the table cell values. Limaye et al. [10] use a graphical model which maps every column header to a class from a known ontology, links table cell values to entities from a knowledge-base and identifies relations between columns. They rely on Yago for background knowledge. The core of our framework is a probabilistic graphical model that captures more semantics, including relations between column headers and between row entities. Current table interpretation systems rely on semantically poor and possibly noisy knowledge-bases and do not attempt to produce a complete interpretation of a table. None generate high quality linked data from the inferred meaning or can interpret columns with numeric values and use the results as evidence in table interpretation, a task essential for many domains.

5 Conclusions

Generating an explicit representation of the meaning implicit in tabular data will support automatic integration and more accurate search. In this paper, we presented an implementation of our graphical model which infers a table's meaning relative to a knowledge base of general and domain-specific knowledge. We described a novel *Semantic Message Passing* algorithm which avoids computing potentially huge joint-probability distribution tables normally required in such graphical models.

A thorough evaluation showed promising results, but leaves room for improvement. We believe our extensible and domain independent framework can address the existing challenges in converting tabular data to RDF or high quality linked data. In the future, we will work on designing a cooperative environment in which a person-in-the-loop identifies bad system choices for column classes and relations, cell value entities and optionally suggests better ones from the alternate candidates.

Acknowledgement. This work was supported by AFOSR award FA9550-08-1-0265 NSF awards 0326460 and 0910838 and a gift from Microsoft Research.

References

1. Bizer, C., Lehmann, J., Kobilarov, G., Auer, S., Becker, C., Cyganiak, R., Hellmann, S.: Dbpedia - a crystallization point for the web of data. Journal of Web Semantics 7(3), 154–165 (2009)

2. Cafarella, M.J., Halevy, A.Y., Wang, Z.D., Wu, E., Zhang, Y.: Webtables: exploring the power of tables on the web. PVLDB 1(1), 538–549 (2008)
3. Ding, L., DiFranzo, D., Graves, A., Michaelis, J.R., Li, X., McGuinness, D.L., Hendler, J.A.: TWC data-gov corpus: incrementally generating linked government data from data.gov. In: Proc 19th WWW, pp. 1383–1386. ACM (2010)
4. Dredze, M., McNamee, P., Rao, D., Gerber, A., Finin, T.: Entity disambiguation for knowledge base population. In: COLING, pp. 277–285 (2010)
5. Embley, D.W., Lopresti, D.P., Nagy, G.: Notes on contemporary table recognition. In: Bunke, H., Spitz, A.L. (eds.) DAS 2006. LNCS, vol. 3872, pp. 164–175. Springer, Heidelberg (2006)
6. Han, L., Finin, T.W., Parr, C.S., Sachs, J., Joshi, A.: RDF123: from Spreadsheets to RDF. In: Sheth, A.P., Staab, S., Dean, M., Paolucci, M., Maynard, D., Finin, T., Thirunarayan, K. (eds.) ISWC 2008. LNCS, vol. 5318, pp. 451–466. Springer, Heidelberg (2008)
7. Hurst, M.: Towards a theory of tables. IJDAR 8(2-3), 123–131 (2006)
8. Koller, D., Friedman, N.: Probabilistic Graphical Models: Principles and Techniques. MIT Press (2009)
9. Langegger, A., Wöß, W.: Xlwrap - querying and integrating arbitrary spreadsheets with SPARQL. In: Bernstein, A., Karger, D.R., Heath, T., Feigenbaum, L., Maynard, D., Motta, E., Thirunarayan, K. (eds.) ISWC 2009. LNCS, vol. 5823, pp. 359–374. Springer, Heidelberg (2009)
10. Limaye, G., Sarawagi, S., Chakrabarti, S.: Annotating and searching web tables using entities, types and relationships. In: Proc. 36th VLDB (2010)
11. Mulwad, V., Finin, T., Joshi, A.: A Domain Independent Framework for Extracting Linked Semantic Data from Tables. In: Ceri, S., Brambilla, M. (eds.) Search Computing. LNCS, vol. 7538, pp. 16–33. Springer, Heidelberg (2012)
12. Mulwad, V., Finin, T., Syed, Z., Joshi, A.: Using linked data to interpret tables. In: Proc. 1st Int. Workshop on Consuming Linked Data, Shanghai (2010)
13. Polfliet, S., Ichise, R.: Automated mapping generation for converting databases into linked data. In: Proc. 9th Int. Semantic Web Conf. (November 2010)
14. Puranik, N.: A Specialist Approach for Classification of Column Data. Master's thesis, University of Maryland, Baltimore County (August 2012)
15. Sahoo, S.S., Halb, W., Hellmann, S., Idehen, K., Thibodeau Jr., T., Auer, S., Sequeda, J., Ezzat, A.: A survey of current approaches for mapping of relational databases to rdf. Tech. rep., W3C (2009)
16. Suchanek, F.M., Abiteboul, S., Senellart, P.: PARIS: Probabilistic Alignment of Relations, Instances, and Schema. PVLDB 5(3), 157–168 (2011)
17. Suchanek, F.M., Kasneci, G., Weikum, G.: Yago: A Core of Semantic Knowledge. In: 16th Int. World Wide Web Conf. ACM Press, New York (2007)
18. Syed, Z., Finin, T.: Creating and Exploiting a Hybrid Knowledge Base for Linked Data. Springer (April 2011)
19. Vavliakis, K.N., Grollios, T.K., Mitkas, P.A.: RDOTE- transforming relational databases into semantic web data. In: 9th Int. Semantic Web Conf. (2010)
20. Venetis, P., Halevy, A., Madhavan, J., Pasca, M., Shen, W., Wu, F., Miao, G., Wu, C.: Recovering semantics of tables on the web. In: Proc. 37th VLDB (2011)
21. Wang, J., Shao, B., Wang, H., Zhu, K.Q.: Understanding tables on the web. Tech. rep., Microsoft Research Asia (2011)
22. Zagari, R., Bianchi-Porro, G., Fiocca, R., Gasbarrini, G., Roda, E., Bazzoli, F.: Comparison of 1 and 2 weeks of omeprazole, amoxicillin and clarithromycin treatment for helicobacter pylori eradication: the hyper study. Gut 56(4), 475 (2007)

Bringing Math to LOD: A Semantic Publishing Platform Prototype for Scientific Collections in Mathematics

Olga Nevzorova, Nikita Zhiltsov, Danila Zaikin, Olga Zhibrik, Alexander Kirillovich, Vladimir Nevzorov, and Evgeniy Birialtsev

Kazan Federal University,
Kremlyovskaya 18 Str., 420008 Kazan, Russia
{onevzoro,nikita.zhiltsov,ksugltronteal,olgazhibrik,
alik.kirillovich,nevzorovvn}@gmail.com,
IgenBir@yandex.ru

Abstract. We present our work on developing a software platform for mining mathematical scholarly papers to obtain a Linked Data representation. Currently, the Linking Open Data (LOD) cloud lacks up-to-date and detailed information on professional level mathematics. To our mind, the main reason for that is the absence of appropriate tools that could analyze the underlying semantics in mathematical papers and effectively build their consolidated representation. We have developed a holistic approach to analysis of mathematical documents, including ontology based extraction, conversion of the article body as well as its metadata into RDF, integration with some existing LOD data sets, and semantic search. We argue that the platform may be helpful for enriching user experience on modern online scientific collections.

Keywords: Linked Data, Ontology Engineering, Ontology Extraction.

1 Introduction

The Linking Open Data (LOD) initiative[1] has recently revealed the added value of representing heterogeneous data from different content providers as a single "cloud" of interconnected objects. The data are loaded and transformed to RDF from various sources including relational databases, web pages, and semi-structured textual documents. The unified structured representation benefits follow-up Linked Data consumers. For example, contemporary semantic search applications like the semantic search engine Sindice[2] or mashup Sig.ma[3] harness the published data to be able to either handle search queries more accurately or aggregate information about entities users are interested in.

[1] http://linkeddata.org
[2] http://sindice.com/
[3] http://sig.ma/

H. Alani et al. (Eds.): ISWC 2013, Part I, LNCS 8218, pp. 379–394, 2013.

At the same time, the LOD cloud lacks up-to-date and detailed data sets on professional level mathematics. Currently, there exist some unofficial data sets that make available information from well-known publishers and online collections in the academic domain including ACM[4], DBLP[5], and CiteSeer[6], as Linked Data. They have contributed a large amount of scientific article metadata to the LOD cloud. However, exposing only article metadata for mathematical papers is palliative, since the primary objects of interest in these documents are formulas and certain parts such as theorems or proofs. In our particular case, we have faced with the requirements of the publishing department at Kazan Federal University, which plans to make publicly available metadata as well as the contents of 1 330 articles of the "Izvestiya Vuzov. Matematika" (IVM, Proceedings of Higher Education Institutions: Mathematics) journal published in 1997-2009. The publisher expects that it will benefit professional researchers and learning students at the university, by providing them opportunities to get access to a knowledge source integrated into the global knowledge base. Thus, our primary goal is to develop a machinery that facilitates the process and, eventually, constructs a new LOD data set having a collection of mathematical scholarly articles.

In the paper, we present our approach of designing and implementing a programming solution to extract a semantic LOD representation of mathematical scholarly papers in a given digital collection. The core of the approach is modeling the given collection of documents as a unified semantic graph. Both the nodes (mathematical knowledge objects) and the edges (relations between them) in it are defined by a set of math-aware vocabularies that specify the logical structure of mathematical documents (theorems, proofs, definitions, formulas etc.) as well as mathematical concepts. In summary, our key contributions are:

- a thorough domain model that includes an ontology of the logical structure of mathematical scholarly papers along with an ontology of mathematical knowledge concepts in Russian/English;
- a language-independent method for extraction of the logical structure elements;
- a method for extraction of mathematical named entities from texts in Russian;
- a method that connects mathematical named entities to symbolic expressions.

The rest of the paper is organized as follows. In Section 2, we meticulously describe our approach for publishing mathematical scholarly papers as Linked Data. Section 3 contains implementation details of the developed prototype. We report on our evaluation experiments in Section 4. Section 5 provides the data set statistics and several use cases. Section 6 gives a brief overview of related work. We conclude and discuss the future work in Section 7.

[4] http://acm.rkbexplorer.com/
[5] http://dblp.rkbexplorer.com/
[6] http://citeseer.rkbexplorer.com/

2 Approach

In this section, we first describe our domain-specific ontologies that provide a vocabulary for extraction methods. Next, we present our solution for NLP and semantic annotation tasks. Finally, we explain our techniques for article metadata extraction and interlinking with existing LOD data sets.

2.1 Domain Model

Mocassin Ontology. The ontology[7] of our Mocassin project[8] aims to capture the semantics of typical structural elements in mathematical scholarly papers. The ontology is a compromise between the semantics of highly formalized models we have seen in the previous works (discussed in Section 6) and facts that can be extracted by automatic methods. Each structural element in Mocassin Ontology represents the finest level of granularity and has its inherent features, such as starting and ending positions, text contents, and functional role. In particular, the ontology defines some ubiquitous document parts, such as theorems, lemmas, proofs, definitions, corollaries etc. Besides, the ontology declares two types of object binary relations – navigational and restricted. The property instances of the first relation type, which is represented by *refersTo* and *dependsOn* relations, tend to occur in mathematical documents when the author points at significant parts of a publication in the form of referential sentences. The part-whole property (*hasPart*) and *followedBy* property belong to the first type too. An example of a relation of the second type is *proves* relation, which occurs between a proof – the only valid element type here – and a statement the proof justifies. In our application, we follow the closed world assumption, and interpret range and domain of a property as constraints.

To add support of structural elements that are common for scientific publications on a wide range of fields, the ontology imports SALT Document Ontology (SDO) [1], an ontology of the rhetorical structure of scholarly publications. Specifically, it defines Section, Figure, and Table classes.

To enable making connections between structural elements and other objects contained by them and described elsewhere, e.g. mathematical named entities extracted from their text contents, we add a specific property – *mentions* – as follows: $mentions(x, y) \rightarrow (DocumentSegment(x) \vee Table(x) \vee Figure(x) \vee Section(x)) \wedge Thing(y)$. Document Segment class is the root of the Mocassin Ontology hierarchy.

The ontology also defines classes to represent several types of mathematical expressions – Mathematical Expression, Variable, and Formula. The datatype property *hasLatexSource* is defined for storing a LaTeX representation of the expression as a string. Yet, for the purpose of connecting formulas to mathematical named entities, there is *hasNotation* property in the ontology: $hasNotation(x, y) \rightarrow Thing(x) \wedge MathematicalExpression(y)$. For example, it enables us to state a fact that an empty set is denoted with $\emptyset$ in a text.

[7] `http://cll.niimm.ksu.ru/ontologies/mocassin`
[8] `http://code.google.com/p/mocassin/`

In addition, the ontology contains logical rules and cardinality axioms. One of the cardinality axioms states that every proof must justify at most one statement. An example logical rule is $dependsOn(x, y) \wedge hasPart(z, y) \rightarrow dependsOn(x, z)$, which e.g. we use to infer dependency between a proof and theorem, if the theorem contains an equation the proof depends on.

The ontology has been developed in OWL2/RDFS languages, which provide rich expressiveness, including cardinality and transitivity, and are also decidable theoretically and practically, for example, by using state-of-the-art reasoners like Pellet and FaCT++, or, to some extent, by in-house reasoners in modern RDF triple stores. A possible use case to exploit this feature is visualization of a dependency graph of theorems in related papers.

OntoMath^{PRO}. *OntoMath^{PRO}* is an applied ontology for automatically processing professional mathematical articles in Russian and English[9]. The ontology defines the concepts commonly used in mathematics as well as the developing and not well established vocabulary (e.g. a term *Bitsadze-Samarsky problem* in differential equations). *OntoMath^{PRO}* covers a wide range of fields of mathematics such as number theory, set theory, algebra, analysis, geometry, mathematical logic, discrete mathematics, theory of computation, differential equations, numerical analysis, probability theory, and statistics. Each class has a textual explanation, Russian and English labels including synonyms.

The terminological sources used during the development are classical textbooks, online resources like Wikipedia and Cambridge Mathematical Thesaurus, scholarly papers from the IVM journal, and personal experience of practicing mathematicians at Kazan Federal University. Thus, we expect that the ontology suffices the expert-level semantics on the fields.

In the ontology, one could distinguish two taxonomies with respect to ISA-relationship – a hierarchy of fields of mathematics and a hierarchy of mathematical knowledge objects. The first one is rather conventional and close to the related part of the Universal Decimal Classification[10]. The top level of the second taxonomy contains concepts of three types: i) basic metamathematical concepts, e.g. Set, Operator, Map, etc; ii) root elements of the concepts related to the particular fields of mathematics, e.g. Element of Probability Theory or Element of Numerical Analysis; iii) common scientific concepts: Problem, Method, Statement, Formula, etc. Due to multiple inheritance, the same class can be a sub-class of several classes. For example, Sparse Grid is a sub-class of both Formula and Element of Theory of Differential Equations.

OntoMath^{PRO} defines three types of object properties:

- a directed relation between a mathematical knowledge object and a field of mathematics (*belongsTo*), e.g. Barycentric Coordinates *belongsTo* Metric Geometry;
- a directed relation of logical dependency between mathematical knowledge objects (*isDefinedBy*), e.g. Christoffel Symbol *isDefinedBy* Connectedness;

[9] `http://cll.niimm.ksu.ru/ontologies/mathematics`
[10] `http://www.udcc.org`

– a symmetric associative relation ("soft dependency") between mathematical knowledge objects (*seeAlso*), e.g. Chebyshev Iterative Method *seeAlso* Numerical Solution of Linear Equation Systems.

$OntoMath^{PRO}$ is developed in OWL-DL/RDFS languages. Numerically, $OntoMath^{PRO}$ contains 3 450 classes, 5 object properties, 3 630 subclass-of property instances, and 1 140 other property instances.

2.2 NLP Annotation

At this stage, we solve a standard task of annotating noun phrases in mathematical texts. In our approach, mathematical expressions are considered as valid parts of noun phrases. That is, they can be prefixes in hyphenated words, e.g. "σ-algebra".

Our solution relies on the "OntoIntegrator" [2], our tool for general-purpose linguistic analysis, which was adapted for peculiarities of mathematical texts, and currently supports only Russian language. It consecutively solves the standard linguistic tasks such as tokenization, sentence splitting, morphological analysis, and noun phrase extraction.

Morphological analysis is based on the Russian grammar dictionary extended with the vocabularies of general and domain-specific abbreviations, and parentheses. The result of the analysis is a grammar markup for words. In addition, homonyms are annotated with a fixed set of grammar annotations.

In Russian, noun phrases (NP) usually consists of the main noun, which we denote as NP.Head, and its left- and right modifiers (NP.Dependent). The relationship between the main noun and its dependent words is syntactical. Constructing noun phrases is described with the rules, which consider the definitive internal structure.

In our case, the main noun can be a noun, a pronominal noun, an abbreviation, a proper noun, a formula, or a citation reference. Among dependent words, there can be adjectives, pronominal adjectives, numerals, participles, adverbs, and prepositions. The noun phrase extraction method seeks noun phrases within a given sentence. Every noun phrase may contain exactly one or several segments, that is, word groups with certain characteristics. Within a segment, all the words are consistent according to their grammar characteristics. If there is more than one segment extracted in the noun phrase, the leftmost segment is considered as the main one and may have arbitrary grammatical characteristics – the case and the number. We assume that the other segments necessarily require the only form – the genitive case of a dependent noun. Gathering segments in a noun phrase is done from the right to the left. While annotating a noun phrase, the NP.Head is distinguished and normalized. The normalized form of the noun phrase is marked with a special "Form" annotation attribute. Math expressions are annotated with special "Math" tags. Further, the annotated noun phrases are used through ontology extraction.

Replacing the current NL processor with a module that supports noun phrase extraction as well as handling math symbols, abbreviations, formulas could switch the language to English as an example. A math-aware extension of the Stanford NLP parser[11] is a promising candidate.

2.3 Semantic Annotation

During this phase, we perform annotating documents in terms of the domain ontologies.

Mining the Logical Structure. Our method [3] receives NLP annotations and extracts structural elements according to the Mocassin and SALT SDO ontologies. This procedure falls into two tasks: (i) recognizing the types of structural elements; (ii) recognizing the semantic relations between them. As a result, the method outputs a semantic graph that contains, on the one hand, structural elements as nodes, each of which is assigned to a particular ontology class or marked "unrecognized" otherwise, and, on the other hand, ontology relation instances as edges. Aside from the object properties, each node has annotations corresponding to its title, text contents, and page numbers in the compiled PDF document. The information may be used in further applications for organizing a convenient navigation through document parts or highlighting more specific relevant search results.

Mathematical Named Entity Extraction. This task is a classification of extracted noun phrases as instances of $OntoMath^{PRO}$ classes, i.e., mathematical named entities (MNEs).

Our extraction method is uncertain and is based on a overlap of words in a noun phrase and ontology labels, respectively. We use Jaccard similarity coefficient as a confidence measure. Therefore, the method implies choosing the threshold value for filtering out wrong matchings. Specifically, given an NP and an ontology class, the confidence score C is defined according to the following rules:

- C ranges from 0 (minimal confidence) to 1 (maximal confidence);
- if the class label does not contain the main word of the NP (NP.Head), then $C = 0$;
- if the length (in terms of word count) of the class label is greater than the length of the NP, then $C = 0$;
- otherwise, C is equal to the Jaccard similarity coefficient for sets of words.

For example, the score between a noun phrase "Sobolev-like space" and a class Sobolev Space is equal to 2/3. On the contrary, the score between "number" and Fermat Number is equal to 0 because of the different lengths, or the score between "integral of the function of square-rooting" and Function of Square-rooting is also equal to 0 due to the different main words in the phrases.

[11] `http://nlp.stanford.edu/software/lex-parser.shtml`

Connecting MNEs to Formulas. We solve the following tasks within a single document:

- parsing mathematical expressions, i.e., detection of variables and seeking their occurrences in mathematical formulas;
- matching mathematical variables with noun phrases.

The method relies on "Math", token, sentence and NP annotations. Regular expressions are used as a main tool during formula analysis. At the beginning, a formula is refined from special markup elements and redundant spaces. Then, the formula is split into separate elements, the delimiters are braces, brackets, operation symbols, punctuation marks, and spaces. The given elements are assigned to specific groups – markup keywords, indices, numbers etc. Each unclassified element is checked additionally on that its starting symbol is not a number, or if the element is in the set of Greek letters. As a result, all the mathematical expressions are divided into three groups – variables, formulas, and auxiliary fragments.

All the variables and formulas are stored in the index, which contains information about occurrences of variables in formulas. We provide an example that illustrates the semantics of such a relationship.

Example 1. Given a text fragment (translated from Russian):

Let $\overline{\alpha}$ be a second fundamental form of n-surface $\overline{M}$, $\overline{\nabla}$ is a Levi-Civita connection of the metric $\overline{g}$. Then, the equality holds:

$$\partial_X df Y - df \overline{\nabla}_X Y = \overline{\alpha}(X, Y).$$

The text fragment contains variables $\overline{\alpha}$, n, $\overline{M}$, $\overline{\nabla}$, $\overline{g}$, and a formula that uses $\overline{\alpha}$ and $\overline{\nabla}$ variables. Implicit bound variables X and Y are defined nowhere in the document, and, therefore, not included into the index. The instances of *hasPart* relation induced from the inclusions are depicted in Figure 1.

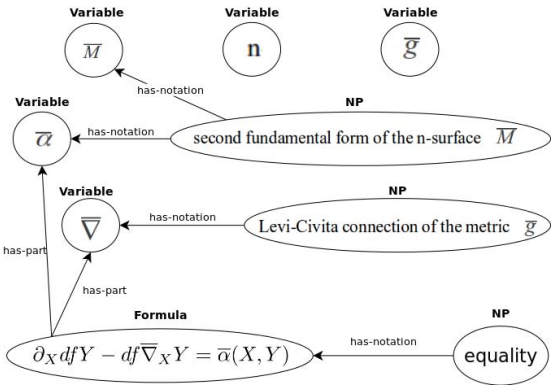


Fig. 1. A semantic graph to Example 1

The next step is connecting noun phrases to extracted variables and formulas. In principle, there are two possible cases of mutual positioning of a variable and an NP: first, an NP may contain a variable, and, second, the elements follow each other.

In the first case, an NP is the only candidate for linking. The simplest variation is if the NP contains a single main word. In Example 1, we have an NP "equality $", where $ is a formula in the NP. This means that the formula will be linked with this NP (see Figure 1). The complex variation is if an NP contains more than one word. In Example 1, variable $\overline{g}$ will not be linked with an NP "Levi-Civita connection of the metric $", because the main word is "Levi-Civita connection" and the variable is a complement here. Similarly, we ignore expression prefixes: in Example 1, "n" is left without linking, but a variable $\overline{M}$ will be linked with an NP "second fundamental form of the $-surface $".

In the second case, the key idea behind analysis is a concept of maximal feasible distance (MFD) in terms of symbol positions between "Math" and NP annotations in the text. For a given pair, we constrain MFD to be always less than the length of a sentence that contains both the annotations. The optimal value for MFD can be found empirically and, as our experiments have shown, the results are robust to its actual value. Finally, the method chooses the closest NP annotation to a given formula. Though, some cases are handled specifically, e.g. such popular text patterns as "[formula] – [NP]" with the dash in the middle.

2.4 Article Metadata Extraction

At this stage, we solve a task of extraction and conversion of article metadata as well as bibliographic references according to a standardized vocabulary. For this purpose, we choose AKT Portal Ontology[12]. Comparing to its alternatives, such as BIBO[13] and SWRC[14], the ontology covers the academic domain in more details and is widely used in existing LOD data sets. The extraction method:

- crawls a collection of documents and extracts from the headers the following information – title, author names, their affiliation, journal title, journal volume, and publication year;
- makes identifiers out of publication titles;
- post-processes bibliographies using the identifiers.
- prepares the article data for for serializing according to the AKT schema.

Article URIs are generated compatible with URLs on MathNet.Ru[15], a large online digital collection. In particular, it means that article URIs from our data set can be easily dereferenced in an Internet browser.

[12] http://www.aktors.org/ontology/
[13] http://bibliontology.com/specification
[14] http://ontoware.org/swrc
[15] http://mathnet.ru

2.5 Interlinking

We solve a task of interlinking the IVM data set with existing data sets in the LOD cloud. Essentially, the task is two-fold: first, aligning $OntoMath^{PRO}$ ontology with DBpedia, and, second, seeking duplicates in the AKT based LOD data sets. Our solution is not integrated with the processing units described above, and, unlike them, requires additional human efforts. We heavily use Silk application[16] for both the subtasks.

Aligning $OntoMath^{PRO}$ with DBpedia. It is based on the following features:

- class and resource labels (*rdfs:label* property);
- links to Wikipedia – during the development of the ontology, some definitions were imported from Wikipedia and refer to it. We compare these references with *foaf:primaryTopic* and *rdfs:labels* property values in DBpedia.

For interlinking, we only use DBpedia resources that belong to the Mathematics category and its subcategories (e.g. Algebra, Geometry, Mathematical logic, Dynamical Systems) up to 5 levels with respect to *skos:broader* property. This is mainly caused by the shortcomings of Silk and DBpedia concerning handling and representing transitive properties[17].

After the linking has been accomplished, we generate triples connecting the classes of the $OntoMath^{PRO}$ with the resources from DBpedia by using *skos:closeMatch* property.

Seeking Duplicates in AKT Based Data Sets. We have investigated data sets based on the AKT schema. It turns out that the CORDIS data set[18] is the only appropriate one at the moment. Matching has been performed using information about organizations. In particular, *akt:name* and *akt:has-pretty-name* properties are used.

3 Implementation

In this section, we provide implementation details of our prototype.

The overall infrastructure of the publishing workflow is depicted in Figure 2. LaTeX is the only input document format supported by the prototype at the moment. Then, we use the arXMLiv tools [4] to convert LaTeX source files into a convenient XML representation. The NLP annotation module is based on the facilities of "OntoIntegrator" [2], a proprietary software tool for linguistic analysis of texts in Russian, developed by two of the authors. It supports XML as an

[16] `http://www4.wiwiss.fu-berlin.de/bizer/silk/`

[17] As we noticed during experiments, using the deeper levels may even lead to poor results. For example, there is a transitive chain between Topology and Alice in Wonderland categories!

[18] `http://cordis.rkbexplorer.com`

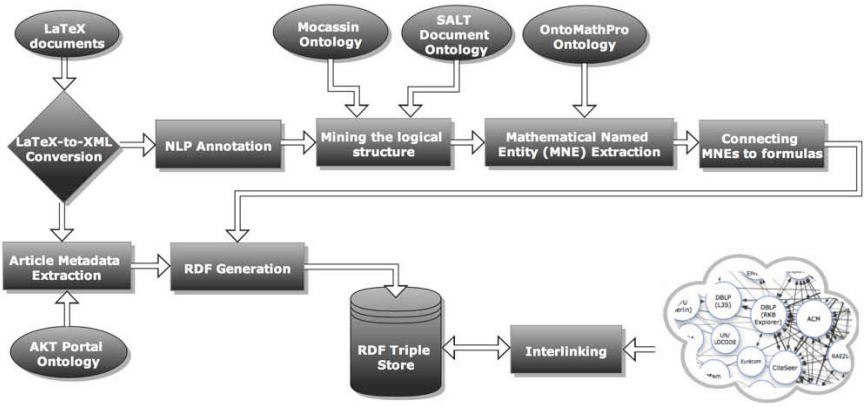


Fig. 2. Prototype Architecture

input/output format. The module for MNE extraction is implemented as a JS script[19]. It accepts XML files for processing and an OWL file of $OntoMath^{PRO}$ ontology. Relying on the NLP annotations, it complements XML files with additional attributes. The module for mining the logical structure is a part of the Mocassin project, an open source mathematical semantic search engine in Java. It processes XML documents using the GATE architecture[20] along with custom processing analyzers. The module for connecting MNEs to formulas is implemented as a GATE plugin[21]. Article metadata extraction is carried out by the special Bash scripts[22]. All the data from the previous steps flow together into the RDF generation unit to be converted to RDF. For the purpose, we use the OpenRDF Sesame library[23] written in Java, which prepares the RDF triple statements and saves them into the triple store, a Virtuoso Community Edition server instance[24]. Virtuoso is a high-performance RDBMS server with extensive RDF/SPARQL support and materialized OWL reasoner. Interlinking is supported by a custom SILK configuration script that uses a list of DBpedia categories related to mathematics[25].

4 Experiments and Evaluation

We have conducted an evaluation of some critical performing tasks to make sure that the extracted data are of high quality. In the section, we present the results and discuss possible failures of the developed methods.

[19] http://bit.ly/cll-mne-extraction
[20] http://gate.ac.uk/
[21] http://bit.ly/cll-gate-morph-formula
[22] http://bit.ly/cll-akt-metadata-extraction
[23] http://www.openrdf.org/
[24] http://sourceforge.net/projects/virtuoso/
[25] http://bit.ly/cll-interlinking

NLP Annotation. We randomly selected 24 documents out of the collection and checked 10 623 NLP annotations assigned by our method. It turns out that the sentence segmentation task is solved at high level of precision (98.9%) and recall (98.83%). Some errors occur, if the author places a period, the mark of the sentence end, inside a mathematical environment. Then, the method for NP extraction gives precision no less than 88%. The error types are as follows: missing fixed prepositional phrases (5%), missing right definition (2%), incomplete NP structure (2%) etc. The method can be improved by more deep syntactical analysis (e.g. of participial phrases) and considering more fixed phrases of mathematical vernacular.

MNE Extraction. While indexing the entire collection, the NLP subsystem outputs 330 462 NPs. The module of MNE extraction links 138 032 (41.7%) NPs to the ontology classes with non-zero confidence score values. After filtering documents on the field of mathematical analysis and removing duplicates, we had 16 300 unique MNE candidates, which were grouped into buckets according to observed confidence score values and were given to an expert in mathematical analysis for manual checking. Table 1 shows the distribution of recall/precision estimates depending on varying the confidence score threshold.

Table 1. Evaluation of MNE Extraction

Confidence score threshold	# of candidates	# of correct candidates	Recall	Precision
0.27	16 300	12 255	1.000	0.752
...	...	...	...	...
0.33	*15 964*	*12 117*	*0.989*	*0.759*
...	...	...	...	...
0.57	**2 470**	**2 426**	**0.198**	**0.982**
...	...	...	...	...
1.00	1 254	1 254	0.102	1.000

Finally, for constructing the RDF data set, we choose the following strategy, which accents the precision:

- for candidates with confidence score greater than 0.57, we generate "hard" relation instances (rdf:type), where every NP is treated as an individual of the linked ontology classes;
- for candidates, which confidence is between 0.33 and 0.57, we generated "soft" relation instances (skos:closeMatch).

Connecting MNEs to Formulas. We have studied the quality of connecting MNEs to formulas depending on the actual value of MFD. We manually select 8 documents from different fields of mathematics. The overall evaluation statistics is shown in Table 2.

Table 2. Statistics of Connecting MNEs to Formulas. TP means true positive, TN – true negative, FP – false positive, FN – false negative.

MFD	TP, %	TN, %	FP, %	FN, %	Accuracy, %
15	36.3	30.5	23.9	9.3	66.8
20	42.3	25.5	25.7	6.5	**67.8**
25	41.0	20.7	23.0	15.3	61.7

In total, there are 1 247 mathematical expressions and 1 357 NPs. The optimal value of MFD is equal to 20. It gives 67.8% in accuracy. We emphasize that this value absorbs the errors of NLP annotation and some misspellings in the texts, e.g. replacing dashes with hyphens. Additionally, varying MFD in a range between 15 and 40 has 64.0% mean accuracy with 2.7% standard deviation, which supports our claim that choosing MFD for our method is not so critical in practice. Among the necessary improvements of the formula linking method, there are a special handling of equation groups and accurately filtering of mathematical expressions.

Aligning *OntoMath^{PRO}* with DBpedia. The alignment has resulted in 947 connections with 907 *OntoMath^{PRO}* classes (some classes were linked with several DBpedia resources). Thus, the ontology coverage is about 27%. The manual assessment gave a precision estimate of 95%. The errors come from the following issues:

- inconsistencies in interwiki linking in Wikipedia: ontomathpro:Sum of the Series ≠ dbpedia:Convergence_tests
- an issue with homonymous concepts and categories in DBpedia: ontomathpro:Ideal ≠ dbpedia:Ideal_ethics, the latter occurs in the transitive chain of categories: Philosophy of Life → Life → Universe → Astronomical dynamical systems → Dynamical Systems.

Seeking Duplicates in AKT Based Data Sets. The module returns only 91 correct and 13 wrong duplicates of organizations from the CORDIS data set. It means that there is no much overlap between these data sets. The module failed to find duplicates of all the types in the DBLP data set due to the absence of such data (in case of organizations) and retrieving limits of its SPARQL endpoint.

5 IVM Data Set: Statistics and Use Cases

The resulting RDF data set[26] contains 854 284 triples including the descriptions of 43 963 variables, 17 397 formulas, 4 190 theorems, 3 035 proofs, 2 356 lemmas, 1

[26] The data set can be accessed via a SPARQL endpoint – `http://cll.niimm.ksu.ru:8890/sparql-auth`, the endpoint is secured, please email the authors to get access to it.

015 definitions and other mathematical entities indexed. Below, we demonstrate several use cases using SPARQL queries to illustrate possible applications.

Use Case 1. Let us assume, we would like to find articles with theorems about finite groups.

```
PREFIX moc: <http://cll.niimm.ksu.ru/ontologies/mocassin#>
PREFIX math: <http://cll.niimm.ksu.ru/ontologies/mathematics#>
SELECT ?article WHERE {
?article moc:hasSegment ?theorem .
?theorem moc:mentions ?entity; a moc:Theorem .
?entity a math:E2183
}
```

In this query, we use Theorem, a Mocassin ontology class, and its properties *hasSegment* and *mentions* along with a class Finite Group (E2183) from the $OntoMath^{PRO}$ ontology.

Use Case 2. The next query is to determine the fields a particular article belongs to.

```
define input:inference
 "http://cll.niimm.ksu.ru/ontologies/mathematics/rules"
PREFIX moc: <http://cll.niimm.ksu.ru/ontologies/mocassin#>
PREFIX math: <http://cll.niimm.ksu.ru/ontologies/mathematics#>
SELECT ?field ?label WHERE {
<http://mathnet.ru/ivm327> moc:hasSegment _:a .
_:a moc:mentions _:b . _:b a _:c .
_:c owl:equivalentClass _:d . _:d owl:onProperty math:P3 ;
owl:allValuesFrom ?field . ?field rdfs:label ?label
} GROUP BY ?field
```

A URI *http://mathnet.ru/ivm327* maps to an article URL on MathNet.Ru. A *math:P3* stands for the inverse property for *belongsTo*. The query outputs classes that represent some mathematical domains, such as Discrete Mathematics, Theory of Computation, Mathematical analysis, and Probability Theory, that are relevant to the given article.

Use Case 3. Finally, for Empty Set, a certain DBpedia concept, we would like to determine its notations occurred in the articles.

```
PREFIX moc: <http://cll.niimm.ksu.ru/ontologies/mocassin#>
SELECT ?latexSource from  <http://cll.niimm.ksu.ru/ivm> WHERE {
?class skos:closeMatch dbpedia:Empty_set .
?notation moc:hasLatexSource ?latexSource .
?entity moc:hasNotation ?notation;
a ?class .
}
```

This query may help to choose the proper notation for a beginning researcher in mathematics. On our data set, the search results are as follows: ω, $\emptyset$, $\omega \in \mathcal{D}$.

6 Related Work

Mathematical knowledge representation, as a field, has its own rich history. There have been developed various models and tools to formalize different aspects of the mathematical domain. For example, domain-specific languages, such as MathLang [5] and OMDoc [6], give opportunities to build semantically enriched models of a mathematical document and natively support representing logical structure elements like theorems or definitions. However, creating such highly formalized mathematical documents is still a laborious process. The paper [7] presented an approach to author math lecture notes with specific sTEX macro package. This work primarily focuses on mathematical formulas and elements of the logical structure and appears to be the first work aiming to fit mathematical texts and LOD together.

Historically, the Bourbaki group's series of books was the first ever attempt to create an ontology of mathematical knowledge rooted in G. Cantor's set theory. Their seminal work establishes a conceptual framework for defining mathematical entities organized in different fields. There have been a few applied domain models developed in the digital era. For example, [8] presents a formal ontology of mathematics for engineers that covers abstract algebra and metrology. Cambridge Mathematical Thesaurus[27] contains a taxonomy of about 4 500 entities connected with logical dependency and associative relationships. This resource covers terms from the undergraduate level mathematics. Next, relying on Wikipedia, Encyclopedia of Science, and the engaged research community, the ScienceWISE project ontology [9] gives over 2 500 mathematical definitions connected with ISA-, part-whole, associative, and importance relationships. The project focuses on achieving a consensus of opinion among mathematicians about given definitions. In the context of modeling mathematical concepts with the help of Semantic Web tools, we would like to note a recent adaptation of Mathematics Subject Classification[28] using SKOS as a linked data set [10]. From this perspective, our $OntoMath^{PRO}$ ontology overlaps with this data set in case of modeling hierarchy of fields, but it is significantly richer for representing mathematical named entities.

Impressive advances in ontology extraction have been achieved across many domains. However, before our work, only a few projects have applied ontology based NLP techniques for scholarly papers in mathematics. The mArachna project [11] focuses on extracting ontologies combining the mathematical knowledge and information about the document structure. However, a comparison of mArachna with our work is problematic, because the project aims for German, and its authors do not provide many details about the specification of the structure, and implementation of the entity extraction techniques to enable a replication of their results. Next, linguistic modules of the arXMLiv project [4] are intended for resolving ambiguities in mathematical notation for texts in English. We are going to conduct a comparative analysis with this work after adding support of English language to our NLP annotation module.

[27] `http://bit.ly/cambridge-math-thesaurus`
[28] `www.ams.org/msc/`

Most research insights and tasks, the solutions of which we described here, were stated in [12]. To our knowledge, the present work is first to extract a Linked Data representation of academic papers in mathematics using not only their metadata, but also the text contents, in an automatic way.

7 Conclusion and Outlook

We present a platform prototype for mining a structured standardized representation of scholarly papers in mathematics. The platform aims for automatic publication their contents as well as metadata in the format of LOD-compliant data. The tool has been applied on a collection of over 1 300 mathematical publications to demonstrate feasibility of the solution. We report on evaluation of the most important tasks solved during the development. Finally, we provide several use cases to illustrate utility of the published data. As a future work, we are aiming to integrate all the modules into a full-fledged toolkit, add support of English language, and extend our approach to other natural science domains, such as physics, chemistry, and biology.

Acknowledgments. This work was supported by the Ministry of Education and Science of the Russian Federation in the scope of the Federal Framework Program – Priority Research and Development in Science and Technology Complex of Russia in 2007-2012 (State contract No. 07.524.11.4005), the Russian Foundation for Basic Research (grant No. 11-07-00507-a), and Kazan Federal University (grant No. 0221 09011). The authors would like to thank Valery Solovyev, Anna Kayumova, Evgeny Lipachev, Ilgiz Kayumov, Pyotr Ivanshin, Elena Utkina, and Marjan Matvejchuk, who have contributed a lot to the $OntoMath^{PRO}$ ontology as well as the conducted evaluation. The authors are also very grateful to Christoph Lange and three anonymous reviewers for their useful suggestions.

References

1. Groza, T., Handschuh, S., Möller, K., Decker, S.: SALT – Semantically Annotated Latex for Scientific Publications. In: Franconi, E., Kifer, M., May, W. (eds.) ESWC 2007. LNCS, vol. 4519, pp. 518–532. Springer, Heidelberg (2007)
2. Nevzorova, O., Nevzorov, V.: The Development Support System "OntoIntegrator" for Linguistic Applications. In: International Book Series "Information Science and Computing", vol. 3(13), pp. 78–84. ITHEA, Rzeszow-Sofia (2009)
3. Solovyev, V., Zhiltsov, N.: Logical Structure Analysis of Scientific Publications in Mathematics. In: Proceedings of the International Conference on Web Intelligence, Mining and Semantics (WIMS 2011), pp. 21:1–21:9. ACM (2011)
4. Stamerjohanns, H., Kohlhase, M., Ginev, D.: Transforming Large Collections of Scientific Publications to XML. In: Mathematics in Computer Science, vol. 3, pp. 299–307. Springer (2010)
5. Kamareddine, F., Wells, J.B.: Computerizing mathematical text with MathLang. Electr. Notes Theor. Comput. Sci., 5–30 (2008)

6. Kohlhase, M.: OMDoc – An Open Markup Format for Mathematical Documents [Version 1.2]. Springer (2006)
7. David, C., Kohlhase, M., Lange, C., Rabe, F., Zhiltsov, N., Zholudev, V.: Publishing Math Lecture Notes as Linked Data. In: Aroyo, L., Antoniou, G., Hyvönen, E., ten Teije, A., Stuckenschmidt, H., Cabral, L., Tudorache, T. (eds.) ESWC 2010, Part II. LNCS, vol. 6089, pp. 370–375. Springer, Heidelberg (2010)
8. Gruber, T., Olsen, G.: An Ontology for Engineering Mathematics. In: KR 1994, pp. 258–269 (1994)
9. Aberer, K., Boyarsky, A., Cudr-Mauroux, P., Demartini, G., Ruchayskiy, O.: ScienceWISE: A Web-based Interactive Semantic Platform for Scientific Collaboration. In: 10th International Semantic Web Conference (ISWC 2011 - Demo) (2011)
10. Lange, C., Ion, P., Dimou, A., Bratsas, C., Sperber, W., Kohlhase, M., Antoniou, I.: Bringing Mathematics to the Web of Data: the Case of the Mathematics Subject Classification. In: Simperl, E., Cimiano, P., Polleres, A., Corcho, O., Presutti, V. (eds.) ESWC 2012. LNCS, vol. 7295, pp. 763–777. Springer, Heidelberg (2012)
11. Jeschke, S., Natho, N., Rittau, S., Wilke, M.: mArachna: Automaticall Extracting Ontologies from Mathematical Natural Language Texts. In: IMECS, pp. 958–963 (2007)
12. Birialtsev, V., Elizarov, A., Zhiltsov, N., Ivanov, V., Nevzorova, O., Solovyev, V.: Ontology Based Semantic Search Model for the Collections of Mathematical Documents. In: Proceedings of XII All-Russian Science Conference RCDL, pp. 296–300 (2010) (in Russian)

ORCHID – Reduction-Ratio-Optimal Computation of Geo-spatial Distances for Link Discovery

Axel-Cyrille Ngonga Ngomo

Department of Computer Science
University of Leipzig
Johannisgasse 26, 04103 Leipzig
ngonga@informatik.uni-leipzig.de
http://limes.sf.net

Abstract. The discovery of links between resources within knowledge bases is of crucial importance to realize the vision of the Semantic Web. Addressing this task is especially challenging when dealing with geo-spatial datasets due to their sheer size and the potential complexity of single geo-spatial objects. Yet, so far, little attention has been paid to the characteristics of geo-spatial data within the context of link discovery. In this paper, we address this gap by presenting ORCHID, a reduction-ratio-optimal link discovery approach designed especially for geo-spatial data. ORCHID relies on a combination of the Hausdorff and orthodromic metrics to compute the distance between geo-spatial objects. We first present two novel approaches for the efficient computation of Hausdorff distances. Then, we present the space tiling approach implemented by ORCHID and prove that it is optimal with respect to the reduction ratio that it can achieve. The evaluation of our approaches is carried out on three real datasets of different size and complexity. Our results suggest that our approaches to the computation of Hausdorff distances require two orders of magnitude less orthodromic distances computations to compare geographical data. Moreover, they require two orders of magnitude less time than a naive approach to achieve this goal. Finally, our results indicate that ORCHID scales to large datasets while outperforming the state of the art significantly.

Keywords: Link discovery, Record Linkage, Deduplication, Geo-Spatial Data, Hausdorff Distances.

1 Introduction

The Linked Open Data Cloud (LOD Cloud) has developed to a compendium of approximately 300 datasets over the last few years. Currently, geographic data sets contain approximately 6 billion triples and make up 19.4% of the triples in the LOD Cloud. Projects such as LinkedGeoData[1] promise an increase of these numbers by orders of magnitude in the near future. However, only 7.1% of the links between knowledge bases in the LOD Cloud currently connect geographic entities. This means that less than 1% of triples within the geographic datasets of the LOD Cloud are links between

[1] See http://linkedgeodata.org. Last access: January 11th, 2013.

H. Alani et al. (Eds.): ISWC 2013, Part I, LNCS 8218, pp. 395–410, 2013.

knowledge bases.[2] This blatant lack of links is partly due to two factors: First, it is due to the *large number of geo-spatial entities* available on the Linked Open Data Cloud. Moreover, the *geo-spatial resources* are often described as (often ordered) sets of points which describe geometric objects such as (multi-) polygons or (multi-) polylines. This way of describing resources differs considerably from the approach followed for most Linked Data resources, which are commonly easiest identified by the means of a label. Consequently, such descriptions have not yet been payed much attention to in the field of link discovery (LD).

We address this gap by presenting ORCHID, a reduction-ratio-optimal approach for LD. ORCHID assumes the LD problem as being formulated in the following way: Given a set S of source instances, a set T of target instances and a distance threshold θ, find the set of triples $(s, t, \delta(s, t)) \in S \times T \times \mathbb{R}^+$ such that $\delta(s, t) \leq \theta$. Given this assumption, the idea behind ORCHID is to address the LD problem on geographic data described as (ordered) sets of points by two means. First, ORCHID implements time-efficient algorithms for computing whether the distance between two polygons s and t is less or equal to a given distance threshold θ. Moreover, ORCHID implements a space tiling algorithm for orthodromic spaces which allows discarding yet another large number of unnecessary computations.

The rest of this paper is structured as follows: In Section 2, we present the core notation used throughout this paper as well as some formal considerations underlying our approach. Section 3 presents two approaches that allow computing the Hausdorff distance between two polygons efficiently.[3] Subsequently, we present the space discretization approach implemented by ORCHID and show that it is optimal with respect to its reduction ratio. We then present a thorough evaluation of our approach on three datasets of different sizes and complexity. We also compare our approach with a state-of-the-art LD framework which implements the orthodromic distance. We conclude the paper with a brief overview of related work (Section 6) and a discussion of our results (Section 7). The approach presented here was integrated in the LIMES framework.[4] Due to space restrictions, we had to omit some details of the approaches presented herein. These can be found in the corresponding technical report on the project webpage.

2 Preliminaries

The formal specification of LD adopted herein is tantamount to the definition proposed in [11]: Given a set S of source resources, a set T of target resources and a relation R, our goal is to find the set $M \subseteq S \times T$ of pairs $(s, t) \in S \times T$ such that $R(s, t)$. If R is `owl:sameAs`, then we are faced with a *deduplication task*. Given that the explicit computation of M is usually a very complex endeavor, M is usually approximated by a set $\tilde{M} = \{(s, t, \delta(s, t)) \in S \times T \times \mathbb{R}^+ : \delta(s, t) \leq \theta\}$, where δ is a distance function and $\theta \geq 0$ is a distance threshold. For geographic data, the resources s and t are described

[2] See `http://wifo5-03.informatik.uni-mannheim.de/lodcloud/state/` for an overview of the current state of the Cloud. Last access: January 11th, 2013.

[3] The Hausdorff distance can be used to compare the distance between any two sets of ordered points located in a space where a distance function is defined. Thus, while we focus on polygons in this paper, our approach can be used for all sets of points.

[4] `http://limes.sf.net`

by using single points or (ordered) sets of points, which we regard as polygons. Given that we can regard points as polygons with one node, we will speak of resources being described as polygons throughout this paper. We will use a subscript notation to label the nodes that make up resources. For example, if s had three nodes, we would denote them s_1, s_2, and s_3. For convenience's sake, we will write $s = \{s_1, s_2, s_3\}$ and $s_i \in s$.

While there are several approaches for computing the distance between two polygons [2], a common approach is the use of the Hausdorff distance [14] *hd*:

$$hd(s,t) = \max_{s_i \in s}\{\min_{t_j \in t}\{\delta(s_i,t_j)\}\}, \tag{1}$$

where δ is the metric associated to the affine space within which the polygons are defined. We assume that the earth is a perfect ball with radius $R = 6378\,km$. Then, δ is the *orthodromic distance* and will be denoted *od* in the rest of this paper. Given these premises, the LD task we investigate in this paper is the following: Find the set $\tilde{M} = \{(s,t,hd(s,t)) \in S \times T : hd(s,t) \leq \theta\}$ where $\forall s_i \in s\ \forall t_j \in t\ \delta(s_i,t_j) = od(s_i,t_j)$. It is important to notice that the orthodromic distance is known to be a metric, leading to the problem formulated above being expressed in a metric space.

Two requirements are central for the approaches developed herein. First, the approaches have to be *complete* (also called lossless [11]), which simply means that they must be able to compute *all triples* $(s,t,hd(s,t)) \in S \times T \times \mathbb{R}^+$ for which $hd(s,t) \leq \theta$ holds. This characteristic is not fulfilled by certain blocking approaches, which trade runtime efficiency for completeness. In addition to developing a complete approach, we aim to develop a reduction-ratio-optimal approach [10]: Let $\mathcal{A}$ be an algorithm for computing $\tilde{M}$ and α be the vector that contains all parameters necessary to run $\mathcal{A}$. Moreover, let $|A(\alpha)|$ be the number of computations of *hd* carried out by $\mathcal{A}$ when assigned the vector of parameters α. We call $\mathcal{A}(\alpha)$ reduction-ratio-optimal when

$$\forall r < 1 - \frac{|\tilde{M}|}{|S||T|}\ \exists \alpha : 1 - \frac{|A(\alpha)|}{|S||T|} \geq r. \tag{2}$$

Naive approaches to computing $\tilde{M}$ have two drawbacks: First, they require $|s||t|$ calls of *od* to compute $hd(s,t)$. Moreover, they carry out $|S||T|$ computations of *hd* to find all elements of $\tilde{M}$. Addressing the time complexity of LD on geographic data thus requires addressing these two quadratically complex problems. Our approach addresses the time complexity of the first problem by making use of the Cauchy-Schwarz inequality, i.e.,

$$od(x,y) \leq od(x,z) + od(z,y), \tag{3}$$

and of bounding circles for approximating the distance between polygons. The second problem is addressed by the means of a reduction-ratio-optimal tiling approach similar to the $\mathcal{HR}^3$ algorithm [10].

3 Efficient Computation of Hausdorff Distances

Several approaches have addressed the time-efficient computation of Hausdorff distances throughout literature (see [14] for a good overview). Yet, so far, these approaches

have not been concerned with the problem of only finding those triples $(s, t, hd(s, t))$ with $hd(s, t) \leq \theta$. In the following, we present several approaches for achieving this goal. These approaches are later evaluated in Section 5. For space reasons, we omit the pseudo-code for the first two approaches. These can be found in the technical report.

3.1 Naive Approach

The naive approach for computing $hd(s, t)$ would compare all elements of the polygon $s \in S$ with all elements of the polygons $t \in T$ by computing the orthodromic distance between all $s_i \in s$ and $t_j \in t$. Let $\bar{S}$ be the average size of the polygons in S and $\bar{T}$ be the average size of the polygons in T. The best- and worst-case runtime complexities of the naive approach are then $O(|S||T|\bar{S}\bar{T})$.

3.2 Bound Approach

A first idea to make use of the bound $hd(s, t) \leq \theta$ on distances lies in the observation that

$$\exists s_i \in S : \min_{t_j \in t}\{od(s_i, t_j)\} > \theta \rightarrow hd(s, t) > \theta \tag{4}$$

This insight allows terminating computations that would not lead to pairs for which $hd(s, t) \leq \theta$ by terminating the computation as soon as a s_i is found that fulfills Eq. (4). In the best case, only one point of each $s \in S$ is compared to all points of $t \in T$ before the computation of $hd(s, t)$ is terminated. Thus, the best-case complexity of the approach is $O(|S||T|\bar{T})$. In the worst case (i.e., in the case that the set of mappings returned is exactly $S \times T$), the complexity of the bound approach is the same as that of the naive approach, i.e., $O(|S||T|\bar{S}\bar{T})$.

3.3 Indexed Approach

The indexed approach combines the intuition behind the bound approach with geometrical characteristics of the Hausdorff distance by using two intuitions. The first intuition is that if the minimal distance between any point of s and any point of t is larger than θ, then $hd(s, t) > \theta$ must hold. Our second intuition makes use of the triangle inequality to approximate the distances $od(s_i, t_k)$. In the following, we present these two intuitions formally. We dub the indexed approach which relies on the second intuition alone *CS* while we call the indexed approach that relies on both intuitions *BC + CS*.

Intuition 1: Bounding Circles. Formally, the first intuition can be expressed as follows:

$$\min_{s_i \in s,\ t_j \in t}\{od(s_i, t_j)\} > \theta \rightarrow hd(s, t) > \theta. \tag{5}$$

Finding the two points s_i and t_j which minimize the value of $od(s_i, t_j)$ requires $O(|s||t|)$ computations of od, i.e., $O(|S||T|\bar{S}\bar{T})$ overall. However, a lower bound for this minimum for all pairs $(s, t) \in S \times T$ can be computed efficiently by using encompassing circles: Let $C(s)$ resp. $C(t)$ be the smallest circles that fully encompass s resp. t. Moreover, let

$r(s)$ resp. $r(t)$ be the radius of these circles and $\zeta(s)$ resp. $\zeta(t)$ be the centers of the circles $C(s)$ resp. $C(t)$. Then,

$$\min_{s_i \in s, t_j \in t} \{od(s_i, t_j)\} > od(\zeta(s), \zeta(t)) - (r(s) + r(t)) = \mu(s, t). \tag{6}$$

Figure 1 displays the intuition behind this approximation graphically. Note that this equation also holds when the circles overlap (in which case $od(\zeta(s), \zeta(t)) - (r(s) + r(t)) < 0$ as $od(\zeta(s), \zeta(t)) < (r(s) + r(t))$.

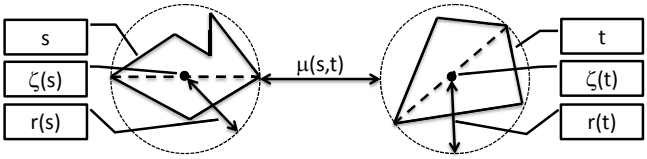


Fig. 1. Lower bound of Hausdorff distances based on circles

Computing the smallest circle that encompasses any polygon x can be carried out in $O(|x|^2)$ by simply computing $od(x_i, x_k)$ for all $(x_i, x_k) \in x^2$. Then,

$$r(x) = \frac{\max_{x_i \in x, x_k \in x} od(x_i, x_k)}{2} \tag{7}$$

while

$$\zeta(x) = \frac{x^+ + x^-}{2} \text{ where } (x^+, x^-) = \arg\max_{x_i \in x, x_k \in x} od(x_i, x_k). \tag{8}$$

The proof that the radius $r(x)$ must have the value shown in Equation 7 is as follows: The points within a circle with radius r' are at most at a distance $2r'$ of each other. Consequently, any circle with radius $r' < r(x)$ cannot contain both elements of the pair $(x^+, x^-) = \arg\max_{x_i \in x, x_k \in x} od(x_i, x_k)$. Thus, the smallest possible radius of a circle that encompasses x fully must be the maximal distance between points which belong to x. This is exactly the value of $r(x)$. Now the only way to ensure that a circle with radius $r(x)$ really encompasses all points in x is to have x^+ and x^- to be diametrically opposite. Thus, $\zeta(x)$ must be exactly in the middle of x^+ and x^-.

The runtime complexity of this approximation is $O(|S|\bar{S}^2 + |T|\bar{T}^2 + |S||T|)$. $O(|S|\bar{S}^2 + |T|\bar{T}^2)$ computations of od are required to determine the circles and their radii while $O(|S||T|)$ computations are required to compare the circles computed out of S with those from T. Note that for large problem $\bar{S}^2$ resp. $\bar{T}^2$ are very small compared to $|S|$ resp. $|T|$, leading to $O(|S|\bar{S}^2 + |T|\bar{T}^2 + |S||T|) \approx O(|S| + |T| + |S||T|) \approx O(|S||T|)$.

Intuition 2: Distance Approximation Using the Cauchy-Schwarz Inequality. Now given that we have computed all distances between all pairs $(t_j, t_k) \in t^2$, we can reuse this information to approximate distances from any s_i to any t_k by relying on the Cauchy-Schwarz inequality in a fashion similar to the LIMES algorithm presented

in [12]. The idea here is that we can compute an upper and a lower bound for the distance $od(s_i, t_k)$ by using the distance $od(s_i, t_j)$ previously computed as follows:

$$|od(s_i, t_j) - od(t_j, t_k)| \leq od(s_i, t_k) \leq od(s_i, t_j) + od(t_j, t_k). \quad (9)$$

For each s_i, exploiting these pre-computed distances can be carried out as follows: For all t_k for which $od(s_i, t_k)$ is unknown, we approximate the distance from s_i to t_k by finding a point t_j for which

$$t_j = \arg\min_{t_x \in t'} od(t_x, t_k) \quad (10)$$

holds, where $t' \subseteq t$ is the set of points t_x of t for which $od(s_i, t_x)$ is known. We call the point t_j an *exemplar* for t_k. The idea behind using one of points closest to t_k is that it gives us the best possible lower bound $|od(s_i, t_j) - od(t_j, t_k)|$ for the distance $od(s_i, t_k)$. Now if $|od(s_i, t_j) - od(t_j, t_k)| > \theta$, then we can discard the computation of the distance $od(s_i, t_k)$ and simply assign it any value $\Theta > \theta$. Moreover, if $|od(s_i, t_j) - od(t_j, t_k)|$ is larger than the current known minimal distance between s_i and points in t, then we can also discard the computation of $od(s_i, t_k)$. If such an exemplar does not exist or if our approximations fail to discard the computation, then only do we compute the real value of the distance $od(s_i, t_k)$.

The best-case complexity of this step alone would be $O(|S||T|\bar{S})$ while in the worst case, we would need to carry out $O(|S||T|\bar{S}\bar{T})$ computations of od. The overall complexity of the indexed approach is $O(|S|\bar{S}^2 + |T|\bar{T}^2 + |S||T|)$ (i.e., that of the bounding circles filter) in the best case and $O(|S|\bar{S}^2 + |T|\bar{T}^2 + |S||T| + |S||T|\bar{S}\bar{T})$ in the worst case. The overall algorithm underlying the indexed approach is shown in Algorithm 1.

4 ORCHID

Although the indexed method presented above can significantly reduce the number of computations carried out to compare S and T, it still needs at least $|S||T|$ comparisons. For example, imagine our source and target data sets were all geo-spatial entities on the portion of the surface of the planet shown in Figure 2. If Oslo (which has the coordinates (59?56'58" N, 10?45'23" E)) was the resource to link via `dbp:near`, then the approaches above would compare it with each of the other elements of the dataset. The idea behind Orchid is to reduce the number of comparisons even further while remaining complete and being reduction-ratio-optimal. To achieve this goal, Orchid uses a space discretization approach and only compares polygons $t \in T$ which lie within a certain range of $s \in S$. An example of the discretization generated by Orchid is shown in Figure 2. Instead of comparing Oslo with all other elements of the dataset, Orchid would only compare it with the geo-spatial objects shown in the gray cells. In the following, we present Orchid formally and prove that it is both complete and reduction-ratio-optimal.

4.1 Preliminaries

Explaining the approach implemented by Orchid prerequisites the explication of a set of characteristics of the orthodromic distance od. Given a polygon s, finding all points

Algorithm 1. Implementation of the $BC + CS$ Hausdorff distance computation. The implementation of CS lacks lines 1,2,3 and 28.

```
 1: if (od(c(s), c(t)) − r(s) − r(t) > θ) then
 2:     return ∅
 3: else
 4:     max ← 0
 5:     for s_i ∈ s do
 6:         min ← ∞
 7:         for t_j ∈ t do
 8:             e = exemplar(t_j)
 9:             if e ≠ ∅ then
10:                 approx = |od(s_i, e) − od(e, t_j)|
11:                 if approx > θ ∨ approx > min then
12:                     d(s_i, t_j) = θ + 1
13:                 else
14:                     d(s_i, t_j) = od(s_i, t_j)
15:                 end if
16:             else
17:                 d(s_i, t_j) = od(s_i, t_j)
18:             end if
19:             min = min(min, d(s_i, t_j))
20:         end for
21:         max = max(max, min)
22:     end for
23:     if max > θ then
24:         return ∅
25:     else
26:         return max
27:     end if
28: end if
```

t such that $hd(s,t) \leq \theta$ requires being able to find all points y for which $od(x,y) \leq \theta$ given a point x. In general, a point x on the surface of the planet can be characterized by two values: its latitude $lat(x)$ and its longitude $long(x)$. These two values are bound as $-\pi/2 \leq lat(x) \leq \pi/2$ and $-\pi \leq lon(x) \leq \pi$ always hold.[5] We write $x = (lat(x), lon(x))$ to denote points. Now, given a point y with $lon(x) = lon(y)$, then $od(x,y) = R|lat(x) - lat(y)|$. Yet, if $lat(x) = lat(y)$, then $od(x,y) = R|lon(x) - lon(y)|cos(lat(x))$. This difference between latitude and longitude is central when finding all points y for which $od(x,y) \leq \theta$. Formally, it means that we can create a discretization in which we treat the latitude values independently from the longitude values but not the other way around. This particular characteristic of latitude and longitude values lies at the heart of Orchid.

4.2 Discretization for Geo-Spatial Points

The idea behind Orchid is to make use of the values of latitude and longitude being bound to first create a grid on the surface of the planet. We call $\alpha \in \mathbb{N}$ the granularity

[5] All angles in this paper are assumed to be in radian unless stated otherwise.

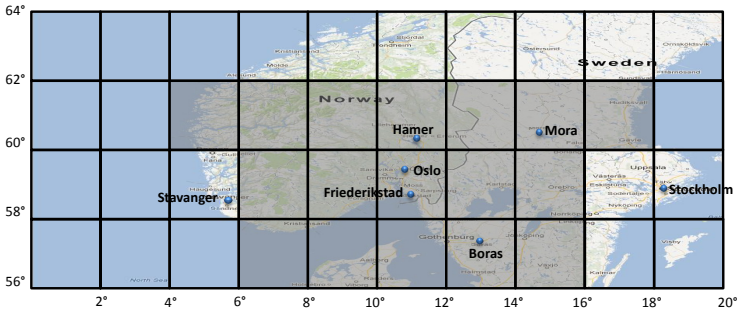


Fig. 2. Example of tiling for $\alpha = 1$ and $\theta = 222.6km$ (i.e., $\Delta_R = 2°$). Here, the resource to link is Oslo. The gray cells are the elements of A(Oslo).

parameter of Orchid. Given the premises described in Section 4.1, we can infer that for $x \in s \in S$ and $t \in T \in T$

$$od(x, y) \leq \theta \rightarrow |lat(x) - lat(y)| \leq \theta/R = \theta_R. \tag{11}$$

Based on this equation, we can create a grid such that width and height of each cell of the grid is $\Delta_R = \theta_R/\alpha$. For each cell c_i, two whole numbers c_i^{lat} and c_i^{lon} exist such that c_i contains only points x for which

$$(c_i^{lat}\Delta \leq lat(x) < (c_i^{lat} + 1)\Delta) \wedge (c_i^{lon}\Delta \leq lon(x) < (c_i^{lon} + 1)\Delta) \tag{12}$$

holds. We call (c_i^{lat}, c_i^{lon}) the coordinates of c_i. Moreover, we write $x \in c_i$ if Equation 12 holds for x. We also write $c_i(x)$ to signify the cell to which x belongs. In our example (Figure 2), the cell which contains Oslo has the coordinates (29, 5). Given this definition of a grid, the set $\tilde{M}(x)$ of y with $od(x, y) \leq \theta$ is clearly a subset of all y for which $|lat(x) - lat(y)| \leq \theta_R$ holds. With respect to our grid, we can infer the following inequality:

$$x \in c_i \wedge y \in c_j \wedge |c_i^{lat} - c_j^{lat}| > \alpha \rightarrow y \notin \tilde{M}(x). \tag{13}$$

We call the set of all cells which abide by this inequation $LAT(x)$. Finding a similar equation for longitudes is more demanding, as the equation depends on the latitude of cells c_i and c_j. Formally, the set $\tilde{M}(x)$ of y with $od(x, y) \leq \theta$ is clearly a subset of all y for which $|lat(x) - lat(y)| \leq \theta_R / \min\{cos(lon(x), lon(y))\}$ holds. Consequently, we can derive the following equation:

$$x \in c_i \wedge y \in c_j \wedge |c_i^{lon} - c_j^{lon}| > \left\lceil \frac{\alpha}{mincos(c_i, c_j)} \right\rceil \rightarrow y \notin \tilde{M}(x) \tag{14}$$

where

$$mincos(c_i, c_j) = \min\{cos(\alpha c_i), cos(\alpha(c_i + 1)), cos(\alpha(c_j)), cos(\alpha(c_j + 1)). \tag{15}$$

We call this set $LON(x)$. Now, if one the minimal cosine values in Equation 15 is 0, then Equation 14 is not well-defined. This happens when one of the cells c_i or c_j is adjacent to

one of the poles. In this case, we assume $\frac{\alpha}{mincos(c_i,c_j)} = 0$. This assumption has the simple consequence that we select all cells c_j at the poles to contain potential y with $od(x, y) \leq \theta$. We can now generate a first approximation of $\tilde{M}(x)$ by computing the intersection of all y that abide by Equations 13 and 14. We call this set $A(x) = LAT(x) \cap LON(x)$. Note that $\tilde{M}(x) \subseteq A(x)$. An example of such a set is shown in Figure 2 where A(Oslo) is depicted as a set of gray squares. Note that given that $\alpha = 1$, we only need to consider the cells with $28 \leq c_i^{lat} \leq 30$. Yet, given that cos(60?) = 0.5, the number of cells that have to be considered in longitude grows from 5 to 7 when crossing the 60th north parallel.

4.3 Optimality of Orchid for Points

While it is guaranteed that $\tilde{M}(x) \subseteq A(x)$, it is possible that $A(x)$ contains grid cell c with $\forall y \in c, (x, y, od(x, y)) \notin \tilde{M}$.[6] Such cells must be eliminated from $A(x)$ as they lead to unnecessary comparisons. Achieving this goal can be carried out by measuring the minimal distance from the cell $c(x)$ which contains x and all other cells $c \in A(x)$. Let us assume that c is at the north east of $c(x)$ (for reasons of symmetry, the argumentation can be extended to all other cells). In our example, such a cell would be that which contains Mora. Then the most north eastern point of $ne(c(x))$ has the coordinates $\Delta(c_i^{lat}(x) + 1, c_i^{lon}(x) + 1)$ while the most south western point of $sw(c)$ of c has the coordinates $\Delta(c_i^{lat}, c_i^{lon})$. Consequently, the minimal distance from points in $c(x)$ to points in c is

$$\min_{od}(c(x), c) = od(\Delta(c_i^{lat}(x) + 1, c_i^{lon}(x) + 1), \Delta(c_i^{lat}(x), c_i^{lon})). \tag{16}$$

We thus define the set $OPT(x) \subseteq A(x)$ as

$$OPT(x) = \{y \in A(x) : \min_{od}(c(x), c(y)) \leq \theta\}. \tag{17}$$

This set is guaranteed not to contain any cell with which elements of $c(x)$ should be compared. Consequently, it is the set of points x and all other elements of $c(x)$ are compared to by Orchid.

$OPT(x)$ is optimal in the sense that

$$\lim_{\alpha \to +\infty} OPT(x) = \tilde{M}(x). \tag{18}$$

This is simply due to $\alpha \to +\infty$ leading to $\Delta \to 0$. In this case, $c(x) = \{x\}$ and $c(y) = \{y\}$. Thus, $\min_{od}(c(x), c(y)) = od(x, y)$ which allows to infer that $OPT(x) = \tilde{M}(x)$ from Equation 17. Note that this proof shows that Orchid fulfills a necessary and sufficient condition to be reduction-ratio-optimal on single points in the sense of [10]. In our example $A(Oslo) = OPT(Oslo)$.

4.4 Comparing Polygons with Orchid

The extension of $OPT(x)$ to polygons is based on the following observation: Given the definition of Hausdorff distances,

$$hd(s, t) \leq \theta \rightarrow \forall s_i \in s \; \exists t_j \in t : od(s_i, t_j) \leq \theta \tag{19}$$

[6] Note that $od(x, y) = hd(x, y)$ for $|x| = |y| = 1$.

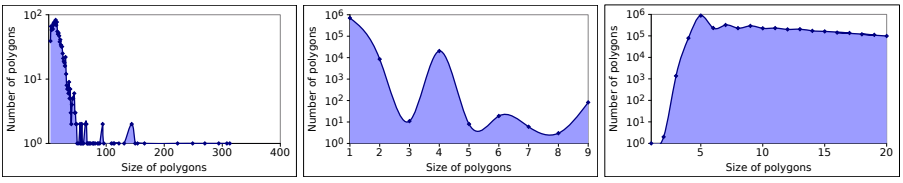
holds. Consequently, $OPT(s) = \bigcap_{s_i \in s} OPT(s_i)$. The reduction-ratio optimality of Orchid for polygons follows from its reduction-ratio-optimality for points.

5 Evaluation

The goal of the evaluation was to assess the performance of our approaches with respect to their runtime and the number of computation of the orthodromic distance that they carried out. To achieve this goal, we first compare the naive, bound, *CS* and *BC+CS* implementations of the computations of bound Hausdorff distance on samples from three different datasets. Note that we refrained from using the whole datasets because the runtime of the naive approach would have been impracticable. In the second part of our evaluation, we study the combination of Orchid and our Hausdorff implementations.

5.1 Experimental Setup

We selected three publicly available datasets of different sizes for our experiments. The first dataset, *Nuts*, contains a detailed description of 1,461 specific European regions.[7] The second dataset, *DBpedia*, contains all 731,922 entries from DBpedia that possess a geometry entry.[8] Finally, the third dataset, LGD, contains all 3,836,119 geo-spatial objects from LinkedGeoData that are instances of the class `Way`.[9] An overview of the distribution of the polygon sizes in these datasets is given in Figure 3. In addition, we used a dataset that consists of all points which have the `wgs84:geometry` property[10] from DBpedia for the comparison with SILK.[11] The 732,224 entities in this dataset are single points on the surface of the planet. We used this dataset because SILK 2.5.3 does not yet support the Hausdorff distance but implements the orthodromic distance.



(a) Nuts **(b)** DBpedia **(c)** LGD

Fig. 3. Distribution of polygon sizes

All experiments were carried out on a 32-core server running JDK 1.7 on Linux 10.04. The processors were 8 quadcore AMD Opteron 6128 clocked at 2.0 GHz. Unless stated otherwise, each experiment was assigned 10GB of memory and was ran 5

[7] We used version 0.9.1 as available at `http://nuts.geovocab.org/data/`

[8] We used version 3.8 as available at `http://dbpedia.org/Datasets`

[9] We used the RelevantWays dataset (version of April 26th, 2011) of LinkedGeoData as available at `http://linkedgeodata.org/Datasets`

[10] `wgs84` stands for `http://www.w3.org/2003/01/geo/wgs84_pos#`

[11] The dataset was extracted from the RelevantNodes dataset (version of April 26th, 2011) of DBpedia as available at `http://linkedgeodata.org/Datasets`

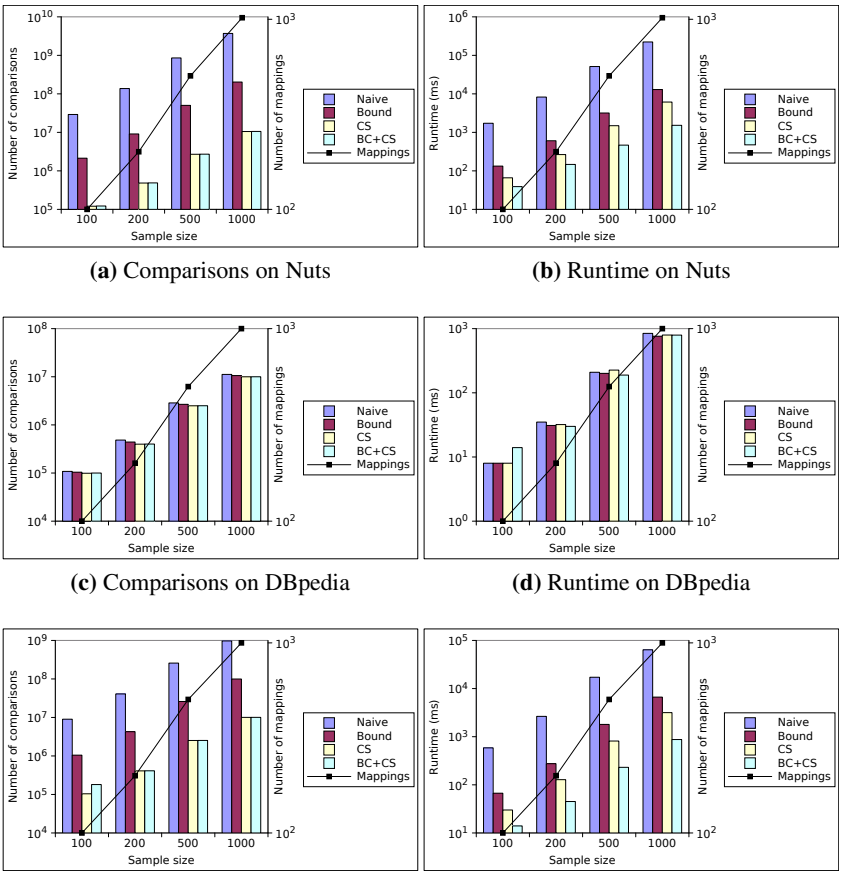
times. The time-out for experiments was set to 3 hours per iteration. The granularity parameter α was set to 1. In the following, we present the minimal runtime of each of the experiments.

5.2 Results

Hausdorff Implementations. In the first part of our evaluation, we measured the runtimes achieved by the three different implementation of the Hausdorff distances on random samples of the Nuts, DBpedia and LGD data sets. We used three different thresholds for our experiments, i.e., 100 m, 0.5 km and 1 km. In Figure 4, we present the results achieved with a threshold of 100 m. The results of the same experiments for 0.5 km and 1 km did not provide us with significantly different insights. All exact values can be found on the project website. As expected the runtime of all three approaches increases quadratically with the size of the sample. There is only a slight variation in the number of comparisons (see Figure 4) carried by the three approaches on the DBpedia dataset. This is simply due to most polygons in the dataset having only a small number of nodes as shown in Figure 3. With respect to runtime, there is no significant difference between the different approaches on DBpedia. This is an important result as it suggests that we can always use the CS or $BC + CS$ approaches even when the complexity of the polygons in the datasets is unknown.

On the two other datasets, the difference between the approaches with respect to both the number of comparisons and the runtime can be seen clearly. Here, the bound implementation requires an order of magnitude less comparisons than the naive approach while the indexed implementations need two orders of magnitude less comparisons. The runtimes achieved by the approaches reflect the observations achieved on the comparisons. In particular, the bound approach is an order of magnitude faster than the naive approach. Moreover, the $BC + CS$ approach outperforms the bound approach by approximately one further order of magnitude. Note that up to approximately 1.07% of the comparisons carried out by $BC + CS$ are the result of the indexing step.

Deduplication. In our second series of experiments, we deduplicated the three datasets at hand by using four different thresholds between 100 m and 2 km. We compared the combination of Orchid ($\alpha = 1$) and of all different implementations of the Hausdorff distance. The rationale behind this experiment was to measure whether the bound and indexed implementations were of any use even within the smaller sub-problems generated by Orchid. The results achieved show that using these implementations can indeed lead to significant improvements in both runtime and comparisons (see Figure 5). In particular, the indexed distance profits from the fact that it can discard a large number of computations that would lead to distance below and above the distance threshold. Thus, it requires over than two orders of magnitude less computations than the bound and naive versions on the Nuts dataset. Given the small size of the index that it generates for Nuts, the indexed approach is also two orders of magnitude faster across all the thresholds. On the LGD dataset, the indexed approach is the only one that terminated within the set time of 3 hours. Due to the topology of the DBpedia data, the runtimes on DBpedia are comparable for all approaches. Here, it is important to note that for smaller thresholds, the indexed approach still requires close to an order of magnitude less comparisons than the naive approach.



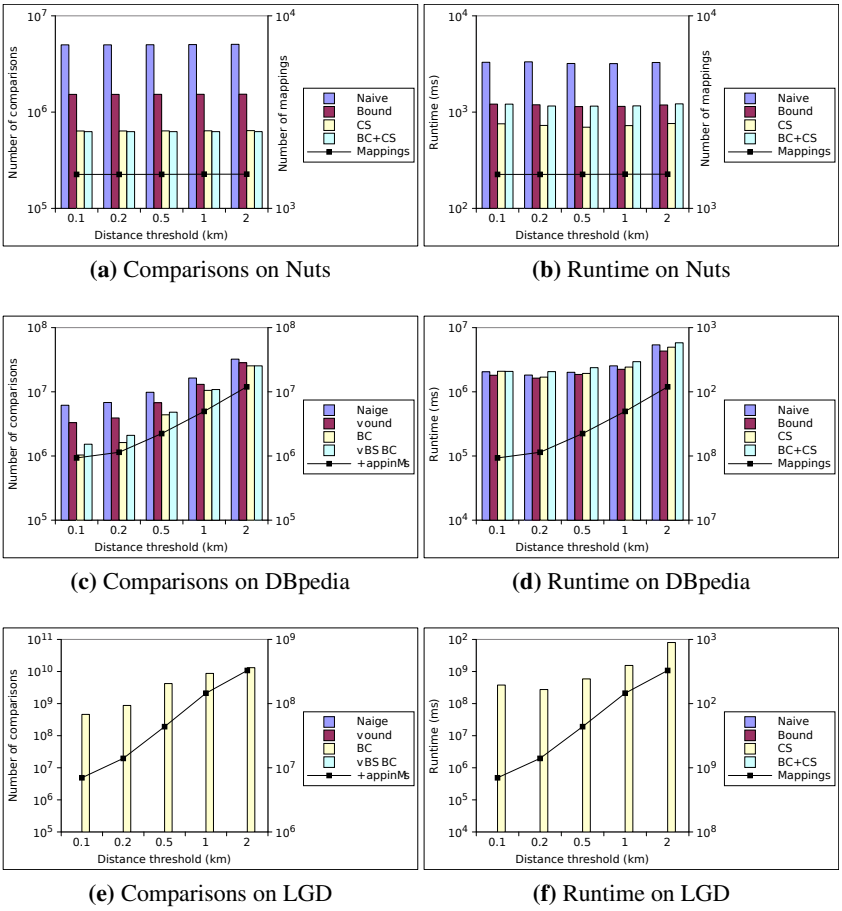
(a) Comparisons on Nuts **(b)** Runtime on Nuts

(c) Comparisons on DBpedia **(d)** Runtime on DBpedia

(e) Comparisons on LGD **(f)** Runtime on LGD

Fig. 4. Number of comparisons and runtimes on samples of the datasets

Scalability. We were also interested in knowing how our approach performs with growing dataset sizes. We thus ran Orchid in combination with *BC* with randomly selected slices of LinkedGeoData and DBpedia and computed the runtime against the size of the data slices. The similarity threshold was set to 0.1 *km* as in the previous experiment. The results on DBpedia and LinkedGeoData are shown in Table 1. We omitted Nuts because it is too small for scalability experiments. The runtimes and number of comparisons on DBpedia suggest that the approach behaves in a quasi-linear fashion on low-dimensional and sparsely distributed data. Note that the number of mappings because partly larger than the number of computations on this dataset is simply due to items with the same URI being found in both source and target and thus not necessitating any comparisons for deduplication. This is more rarely the case in the LinkedGeoData dataset. The runtimes on LinkedGeoData yet suggest that both the number of computations and the runtime required of our approach grow sub-linearly with the number of mappings to be computed when the number of points per polygon grows. This can be explained



(a) Comparisons on Nuts **(b)** Runtime on Nuts

(c) Comparisons on DBpedia **(d)** Runtime on DBpedia

(e) Comparisons on LGD **(f)** Runtime on LGD

Fig. 5. Number of comparisons and runtime of Orchid

by our approach making effective use of existing data to discard computations and reduce the ratio of number of computations to mappings with growing data size. Thus, our approach promises to scale well to even larger data sets.

Comparison with Other Approaches. SILK[12] [6] is of the few other LD framework which implements the orthodromic distance. To the best of our knowledge, no other LD framework implements the Hausdorff distance. Thus, we compare Orchid in combination with the naive implementation of the Hausdorff distance to SILK on all 732,224 points from DBpedia that contain longitude and latitude information. The results of four different distance thresholds are shown in Figure 6. Our results clearly show that Orchid outperforms SILK by more than one order of magnitude in all settings.

[12] Throughout our experiments, we used SILK 2.5.3.

Table 1. Scalability results. The top section shows the results on DBpedia while the lower section shows the results on LinkedGeoData.

Sample Size	*od* computations	Runtime (ms)	Mappings
10^5	34,959	2,936	103,428
2×10^5	97,798	5,783	215,096
4×10^5	341,986	10,423	459,681
7.3×10^5	1,035,222	20,727	932,848
10^5	5,703,683	42,437	77,003
2×10^5	11,734,609	57,935	159,878
4×10^5	24,844,435	153,174	342,477
8×10^5	55,212,459	411,248	777,826
16×10^5	131,405,064	819,636	1,902,803

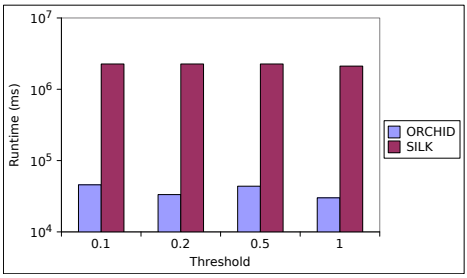


Fig. 6. Comparison of runtime of SILK and ORCHID

6 Related Work

The work presented herein is related to record linkage, deduplication, LD and the efficient computation of Hausdorff distances. An extensive amount of literature has been published by the database community on record linkage (see [7,4] for surveys). With regard to *time complexity*, time-efficient deduplication algorithms such as PPJoin+ [19], EDJoin [18], PassJoin [8] and TrieJoin [17] were developed over the last years. Several of these were then integrated into the hybrid LD framework LIMES [11]. Moreover, dedicated time-efficient approaches were developed for LD. For example, RDF-AI [15] implements a five-step approach that comprises the preprocessing, matching, fusion, interlink and post-processing of data sets. [12] presents an approach based on the Cauchy-Schwarz that allows discarding a large number of unnecessary computations. The approaches HYPPO [9] and $\mathcal{HR}^3$ [10] rely on space tiling in spaces with measures that can be split into independent measures across the dimensions of the problem at hand. Especially, $\mathcal{HR}^3$ was shown to be the first approach that can achieve a relative reduction ratio r' less or equal to any given relative reduction ratio $r > 1$. Standard blocking approaches were implemented in the first versions of SILK and later replaced with MultiBlock [6], a lossless multi-dimensional blocking technique. KnoFuss [13] also implements blocking techniques to achieve acceptable runtimes.

Hausdorff distances are commonly used in fields such as object modeling, computer vision and object tracking. [1] presents an approach for the efficient computation of Hausdorff distances between convex polygons. While the approach is quasi-linear in the number of nodes of the polygons, it cannot deal with non-convex polygons as commonly found in geographic data. [5] presents an approach for the comparison of 3D models represented as triangular meshes. The approach is based on a subdivision sampling algorithm that makes use of octrees to approximate the distance between objects. [16] present a similar approach that allows approximating Hausdorff distances within a certain error bound while [3] presents an exact approach. [14] present an approach to compute Hausdorff distances between trajectories using R-trees within an L_2-space. Note that our approach is tailored to run in orthodromic spaces. Still, some of the insights presented in [14] may be usable in an orthodromic space. To the best of our knowledge, none of the approaches proposed before address the problem of finding pairs of polygons (A, B) such that $hd(A, B) \leq \theta$ in an orthodromic space.

7 Conclusion and Future Work

In this paper, we presented Orchid, a LD approach for geographic data. Our approach is based on the combination of Hausdorff and orthodromic distances. We devised two approaches for computing bound Hausdorff distances and compared these approaches with the naive approach. Our experiments showed that we can be more than two orders of magnitude faster on typical geographic datasets such as Nuts and LinkedGeoData. We then presented the space tiling approach which underlies Orchid and proved that it is reduction-ratio-optimal. Our most interesting result was that our approach seems to be sub-linear with respect to the number of comparisons and the runtime it requires. This behavior can be explained by the approach making use of the higher data density to perform better distance approximations and thus discarding more computations of the orthodromic distance. In addition to comparing different parameter settings of Orchid with each other, we also compared our approach with the state-of-the-art LD framework SILK. Our results show that we outperform the blocking approach it implements by more than one order of magnitude. In future work, we will extend our approach by implementing it in parallel and integrating it with a load balancing approach.

Acknowledgement. The work presented in this paper was financed by the EU-FP7 Project GeoKnow (Grant Agreement No. 318159).

References

1. Atallah, M.J.: A linear time algorithm for the hausdorff distance between convex polygons. Technical report, Purdue University, Department of Computer Science (1983)
2. Atallah, M.J., Ribeiro, C.C., Lifschitz, S.: Computing some distance functions between polygons. Pattern Recognition 24(8), 775–781 (1991)
3. Bartoň, M., Hanniel, I., Elber, G., Kim, M.-S.: Precise hausdorff distance computation between polygonal meshes. Comput. Aided Geom. Des. 27(8), 580–591 (2010)

4. Elmagarmid, A.K., Ipeirotis, P.G., Verykios, V.S.: Duplicate record detection: A survey. IEEE Trans. Knowl. Data Eng. 19(1), 1–16 (2007)
5. Guthe, M., Borodin, P., Klein, R.: Fast and accurate hausdorff distance calculation between meshes. J. of WSCG 13, 41–48 (2005)
6. Isele, R., Jentzsch, A., Bizer, C.: Efficient Multidimensional Blocking for Link Discovery without losing Recall. In: WebDB (2011)
7. Köpcke, H., Rahm, E.: Frameworks for entity matching: A comparison. Data Knowl. Eng. 69(2), 197–210 (2010)
8. Li, G., Deng, D., Wang, J., Feng, J.: Pass-join: a partition-based method for similarity joins. Proc. VLDB Endow. 5(3), 253–264 (2011)
9. Ngonga Ngomo, A.C.: A Time-Efficient Hybrid Approach to Link Discovery. In: OM 2011 (2011)
10. Ngonga Ngomo, A.-C.: Link discovery with guaranteed reduction ratio in affine spaces with minkowski measures. In: Cudré-Mauroux, P., Heflin, J., Sirin, E., Tudorache, T., Euzenat, J., Hauswirth, M., Parreira, J.X., Hendler, J., Schreiber, G., Bernstein, A., Blomqvist, E. (eds.) ISWC 2012, Part I. LNCS, vol. 7649, pp. 378–393. Springer, Heidelberg (2012)
11. Ngonga Ngomo, A.-C.: On link discovery using a hybrid approach. J. Data Semantics 1(4), 203–217 (2012)
12. Ngonga Ngomo, A.-C., Auer, S.: LIMES - A Time-Efficient Approach for Large-Scale Link Discovery on the Web of Data. In: IJCAI, pp. 2312–2317 (2011)
13. Nikolov, A., d'Aquin, M., Motta, E.: Unsupervised learning of link discovery configuration. In: Simperl, E., Cimiano, P., Polleres, A., Corcho, O., Presutti, V. (eds.) ESWC 2012. LNCS, vol. 7295, pp. 119–133. Springer, Heidelberg (2012)
14. Nutanong, S., Jacox, E.H., Samet, H.: An incremental hausdorff distance calculation algorithm. Proc. VLDB Endow. 4(8), 506–517 (2011)
15. Scharffe, F., Liu, Y., Zhou, C.: Rdf-ai: an architecture for rdf datasets matching, fusion and interlink. In: Proc. IJCAI 2009 Workshop on Identity, Reference, and Knowledge Representation (IR-KR), Pasadena, CA, US (2009)
16. Tang, M., Lee, M., Kim, Y.J.: Interactive hausdorff distance computation for general polygonal models. ACM Trans. Graph. 28(3), 74:1–74:9 (2009)
17. Wang, J., Li, G., Feng, J.: Trie-join: Efficient trie-based string similarity joins with edit-distance constraints. PVLDB 3(1), 1219–1230 (2010)
18. Xiao, C., Wang, W., Lin, X.: Ed-Join: an efficient algorithm for similarity joins with edit distance constraints. PVLDB 1(1), 933–944 (2008)
19. Xiao, C., Wang, W., Lin, X., Yu, J.X.: Efficient similarity joins for near duplicate detection. In: WWW, pp. 131–140 (2008)

Simplifying Description Logic Ontologies

Nadeschda Nikitina[1] and Sven Schewe[2]

[1] University of Oxford, UK
[2] University of Liverpool, UK

Abstract. We discuss the problem of minimizing TBoxes expressed in the lightweight description logic $\mathcal{EL}$, which forms a basis of some large ontologies like SNOMED, Gene Ontology, NCI and Galen. We show that the minimization of TBoxes is intractable (NP-complete). While this looks like a bad news result, we also provide a heuristic technique for minimizing TBoxes. We prove the correctness of the heuristics and show that it provides optimal results for a class of ontologies, which we define through an acyclicity constraint over a reference relation between equivalence classes of concepts. To establish the feasibility of our approach, we have implemented the algorithm and evaluated its effectiveness on a small suite of benchmarks.

1 Introduction

It is well-known that the same facts can be represented in many different ways, and that the size of these representations can vary significantly. This is also reflected in ontology engineering, where the syntactic form of ontologies can be more complex than necessary. For instance, throughout the development (and the life-cycle) of an ontology, the way in which concepts and the relationship between them are represented within the ontology are constantly changing. For example, a name for a complex concept expression is often introduced only after it has been used several times and has proved to be important. Another example are dependencies between concepts that evolve over time, resulting in new subsumption relations between concepts ($A_1 \sqsubseteq A_2$). As a result, previously reasonable concept expressions may become unnecessarily complex. In the given example, $A_1 \sqcap A_2$ becomes equivalent to A_1.

Clearly, unnecessary complexity impacts on the maintenance effort as well as the usability of ontologies. For instance, keeping track of dependencies between complex concept expressions and relationships between them is more cumbersome when it contains unnecessarily complex or unnecessarily many different concept expressions. As a result, the chance of introducing unwanted consequences is higher. Moreover, unintended redundancy decreases the overall quality of the ontology.

Removing unnecessary syntactic complexity from ontologies by hand is a difficult task: for the average ontology, it is almost impossible to obtain the minimal representation without tool support. Thus, automated methods that help to assess the current succinctness of an ontology and generate suggestions on how to increase it would be highly valued by ontology engineers.

It is easy to envision scenarios that demonstrate the usefulness of rewriting for reducing the cognitive complexity of axioms. For instance, when a complex concept C is

H. Alani et al. (Eds.): ISWC 2013, Part I, LNCS 8218, pp. 411–426, 2013.

frequently used in the axioms of an ontology and there is an equivalent atomic concept A_C, the ontology will diminish in size when occurrences of C are replaced by A_C.

Example 1. Consider the following excerpt from the ontology Galen [1]:

$$\begin{aligned} \texttt{Clotting} \sqsubseteq\ & \exists\,\texttt{actsSpecificallyOn}.(\texttt{Blood} \sqcap \exists\,\texttt{hasPhysicalState}. \\ & \qquad (\texttt{PhysicalState} \sqcap \exists\,\texttt{hasState.liquid})) \sqcap \\ & \exists\texttt{hasOutcome}.(\texttt{Blood} \sqcap \exists\,\texttt{hasPhysicalState.solidState}) \end{aligned} \tag{1}$$

$$\texttt{LiquidState} \equiv \texttt{PhysicalState} \sqcap \exists\,\texttt{hasState.liquid} \tag{2}$$

$$\texttt{LiquidBlood} \equiv \texttt{Blood} \sqcap \exists\,\texttt{hasPhysicalState.LiquidState} \tag{3}$$

Given concepts defined in Axioms 2 and 3 above, we can easily rewrite Axiom 1 to obtain the following, simpler axiom containing only 6 references to concepts and roles (as opposed to 10 references in Axiom 1):

$$\begin{aligned} \texttt{Clotting} \sqsubseteq\ & \exists\,\texttt{actsSpecificallyOn.LiquidBlood} \sqcap \\ & \exists\texttt{hasOutcome}.(\texttt{Blood} \sqcap \exists\,\texttt{hasPhysicalState.solidState}) \end{aligned} \tag{4}$$

In description logics [2], few results towards simplifying ontologies have been obtained so far. Grimm et al. [3] propose an algorithm for eliminating semantically redundant axioms from ontologies. In the above approach, axioms are considered as atoms that cannot be split into parts or changed in any other way. With the specific goal of improving reasoning efficiency, Bienvenu et al. [4] propose a normal form called prime implicates normal form for $\mathcal{ALC}$ ontologies. However, as a side-effect of this transformation, a doubly-exponential blowup in concept size can occur.

In this paper, we investigate the succinctness for the lightweight description logic $\mathcal{EL}$. The tractable OWL 2 EL profile [5] of the W3C-specified OWL Web Ontology Language [6] is based on DLs of the $\mathcal{EL}$ family [7]. We consider the problem of finding a minimal equivalent representation for a given $\mathcal{EL}$ ontology. First, we demonstrate that we can reduce the size of a representation by up to an exponent even in the case that the ontology does not contain any redundant axioms. We show that the related decision problem (is there an equivalent ontology of size $\leq k$?) is NP-complete by a reduction from the set cover problem, which is one of the standard NP-complete problems. We also show that, just as for other reasoning problems in $\mathcal{EL}$, ontology minimization becomes simpler under the absence of a particular type of cycles. We identify a class of TBoxes, for which the problem can be solved in PTIME instead of NP and implement a tractable algorithm that computes a minimal TBox for this class of TBoxes. The algorithm can also be applied to more expressive and most cyclic TBoxes[1], however without a guarantee of minimality. We apply an implementation of the algorithm to various existing ontologies and show that their succinctness can be improved. For instance, in case of Galen, we managed to reduce the number of complex concepts occurrences by 955 and the number of references to atomic concepts and roles by 1130.

The paper is organized as follows: In Section 2, we recall the necessary preliminaries on description logics. Section 3 demonstrates the potential of minimization. In the

[1] The extension to general TBoxes is a trivial modification of the algorithm.

same section, we also introduce the basic definitions of the size of ontologies and formally state the corresponding decision problem. In Section 4, we derive the complexity bounds for this decision problem. Section 5 defines the class of TBoxes, for which the problem can be solved in PTIME instead of NP and presents a tractable algorithm that computes a minimal TBox for this class of TBoxes. In Section 6, we present experimental results for a selection of ontologies. Finally, we discus related approaches in Section 7 before we conclude and outline future work in Section 8. Further details and proofs can be found in the extended version of this paper.

2 Preliminaries

We recall the basic notions in description logics [2] required in this paper. Let N_C and N_R be countably infinite and mutually disjoint sets of concept symbols and role symbols. An $\mathcal{EL}$ concept C is defined as

$$C ::= A|\top|C \sqcap C|\exists r.C,$$

where A and r range over N_C and N_R, respectively. In the following, we use symbols A, B to denote atomic concepts and C, D, E to denote arbitrary concepts. A *terminology* or *TBox* consists of *concept inclusion* axioms $C \sqsubseteq D$ and *concept equivalence* axioms $C \equiv D$ used as a shorthand for $C \sqsubseteq D$ and $D \sqsubseteq C$. The signature of an $\mathcal{EL}$ concept C or an axiom α, denoted by $\mathrm{sig}(C)$ or $\mathrm{sig}(\alpha)$, respectively, is the set of concept and role symbols occurring in it. To distinguish between the set of concept symbols and the set of role symbols, we use $\mathrm{sig}_C(C)$ and $\mathrm{sig}_R(C)$, respectively. The signature of a TBox $\mathcal{T}$, in symbols $\mathrm{sig}(\mathcal{T})$ (correspondingly, $\mathrm{sig}_C(\mathcal{T})$ and $\mathrm{sig}_R(\mathcal{T})$), is defined analogously. Additionally, we denote the set of subconcepts occurring in a concept C as $\mathtt{sub}(C)$ and the set of all subconcepts including part-conjunctions as $\mathtt{sub}_\sqcap(C)$. For instance, for $C = \exists r.(A_1 \sqcap A_2 \sqcap A_3)$ we obtain $\mathtt{sub}(C) = \{\exists r.(A_1 \sqcap A_2 \sqcap A_3), A_1 \sqcap A_2 \sqcap A_3, A_1, A_2, A_3\}$ and $\mathtt{sub}_\sqcap(C) = \{\exists r.(A_1 \sqcap A_2 \sqcap A_3), A_1 \sqcap A_2 \sqcap A_3, A_1 \sqcap A_2, A_1 \sqcap A_3, A_2 \sqcap A_3, A_1, A_2, A_3\}$. Accordingly, we denote the set of subconcepts occurring in a TBox $\mathcal{T}$ as $\mathtt{sub}(\mathcal{T})$ and the set of all subconcepts including part-conjunctions as $\mathtt{sub}_\sqcap(\mathcal{T})$.

Next, we recall the semantics of the above introduced DL constructs, which is defined by means of interpretations. An interpretation $\mathcal{I}$ is given by the domain $\Delta^\mathcal{I}$ and a function $\cdot^\mathcal{I}$ assigning each concept $A \in N_C$ a subset $A^\mathcal{I}$ of $\Delta^\mathcal{I}$ and each role $r \in N_R$ a subset $r^\mathcal{I}$ of $\Delta^\mathcal{I} \times \Delta^\mathcal{I}$. The interpretation of $\top$ is fixed to $\Delta^\mathcal{I}$. The interpretation of an arbitrary $\mathcal{EL}$ concept is defined inductively, i.e., $(C \sqcap D)^\mathcal{I} = C^\mathcal{I} \cap D^\mathcal{I}$ and $(\exists r.C)^\mathcal{I} = \{x \mid (x, y) \in r^\mathcal{I}, y \in C^\mathcal{I}\}$. An interpretation $\mathcal{I}$ satisfies an axiom $C \sqsubseteq D$ if $C^\mathcal{I} \subseteq D^\mathcal{I}$. $\mathcal{I}$ is a model of a TBox, if it satisfies all of its axioms. We say that a TBox $\mathcal{T}$ entails an axiom α (in symbols, $\mathcal{T} \models \alpha$), if α is satisfied by all models of $\mathcal{T}$. A TBox $\mathcal{T}$ entails another TBox $\mathcal{T}'$, in symbols $\mathcal{T} \models \mathcal{T}'$, if $\mathcal{T} \models \alpha$ for all $\alpha \in \mathcal{T}'$. $\mathcal{T} \equiv \mathcal{T}'$ is a shortcut for $\mathcal{T} \models \mathcal{T}'$ and $\mathcal{T}' \models \mathcal{T}$.

3 Reducing the Complexity of Ontologies

The size of a TBox $\mathcal{T}$ is often measured by the number of axioms contained in it ($|\mathcal{T}|$). This is, however, a simplified view of the size, which neither reflects cognitive

complexity, nor the reasoning complexity. In this paper, we measure the size of a concept, an axiom, or a TBox by the number of references to signature elements as stated in the definition below.

Definition 1. *The* size *of an $\mathcal{EL}$ concept D is defined as follows:*

- *for $D \in sig(\mathcal{T}) \cup \{\top\}$, $\int(D) = 1$;*
- *for $D = \exists r.C$, $\int(D) = \int(C) + 1$ where $r \in sig_R(\mathcal{T})$ and C is an arbitrary concept;*
- *for $D = C_1 \sqcap C_2$, $\int(D) = \int(C_1) + \int(C_2)$ where C_1, C_2 are arbitrary concepts;*

The size *of an $\mathcal{EL}$ axiom (one of $C_1 \sqsubseteq C_2$, $C_1 \equiv C_2$) and a TBox $\mathcal{T}$ is accordingly defined as follows:*

- *$\int(C_1 \sqsubseteq C_2) = \int(C_1) + \int(C_2)$ for concepts C_1, C_2;*
- *$\int(C_1 \equiv C_2) = \int(C_1) + \int(C_2)$ for concepts C_1, C_2.*
- *$\int(\mathcal{T}) = \sum_{\alpha \in \mathcal{T}} \int(\alpha)$ for a TBox $\mathcal{T}$.*

The above definition, for instance, can serve as a basis for computing the average size of axioms ($\int(\mathcal{T}) \div |\mathcal{T}|$) within an ontology. In addition to the above measure of size, the number of distinct complex concept expressions $\mathtt{sub}(\mathcal{T})$ and the overall number of occurrences of such concept expressions (with the corresponding values related to $|\mathcal{T}|$) can serve as an indication of how complex are concept expressions within the ontology. In the following example, we demonstrate the difference between the two measures $|\mathcal{T}|$ and $\int(\mathcal{T})$ and show how the complexity of an ontology can be reduced in principle (by up to an exponent for ontologies without redundant axioms, i.e., axioms that can be omitted without losing any logical consequences).

Example 2. Let concepts C_i be inductively defined by $C_0 = A$, $C_{i+1} = \exists r.C_i \sqcap \exists s.C_i$. Intuitively, C_i of concepts have the shape of binary trees with exponentially many leaves. Clearly, the concepts grow exponentially with i, since $\int(C_i) = 2 + 2 \cdot \int(C_{i-1})$. For a natural number n, consider the TBox $\mathcal{T}_n$:

$$\begin{aligned} C_{n-1} &\sqsubseteq B \\ B_i &\equiv C_i \qquad 1 \leq i \leq n-1 \end{aligned}$$

While $\mathcal{T}_n$ does not contain any redundant axioms, it can easily be represented in a more compact way by recursively replacing each C_i by the corresponding B_i, yielding $\mathcal{T}'_n$:

$$\begin{aligned} B_{n-1} &\sqsubseteq B \\ B_1 &\equiv C_1 \\ B_{i+1} &\equiv \exists r.B_i \sqcap \exists s.B_i \qquad 1 \leq i \leq n-1 \end{aligned}$$

While the number of axioms is the same in both cases, the complexity of $\mathcal{T}_n$ is clearly lower. E.g., for $n = 5$, we obtain $\int(\mathcal{T}_n) = 134$ and $\int(\mathcal{T}'_n) = 24$.

We now consider the problem of finding the minimal equivalent $\mathcal{EL}$ representation for a given TBox. The corresponding decision problem can be formulated as follows:

Definition 2 (P1). *Given an $\mathcal{EL}$ TBox $\mathcal{T}$ and a natural number k, is there an $\mathcal{EL}$ TBox $\mathcal{T}'$ with $\int(\mathcal{T}') \leq k$ such that $\mathcal{T}' \equiv \mathcal{T}$.*

In general, the corresponding minimal result is not unique. We denote the set $\{\mathcal{T}' \mid \mathcal{T}' \equiv \mathcal{T}\}$ by $[\mathcal{T}]$. Note that the minimality of the result is trivially checked by deciding **P1** for a decreasing number k until the answer is negative.

In literature, there are different variations of the ontology minimization problem that cover specific cases. Perhaps the simplest examples for avoidable non-succinctness are axioms that follow from other axioms and that can be removed from the ontology without losing any logical consequences. While some axioms including the last axiom in the above example can be removed in any representations, in general, subsets of axioms can be exchangeable.

Example 3. Consider the ontology $\mathcal{T}$:

$$\begin{array}{cc} C \sqsubseteq \exists r.C & \exists r.D \sqsubseteq D \\ C \sqsubseteq D & \exists r.C \sqsubseteq \exists r.D \end{array}$$

$\mathcal{T}$ has two subset ontologies, $\mathcal{T}_1$ and $\mathcal{T}_2$:

$$\mathcal{T}_1 = \{C \sqsubseteq \exists r.C, \exists r.C \sqsubseteq \exists r.D, \exists r.D \sqsubseteq D\}$$

$$\mathcal{T}_2 = \{C \sqsubseteq \exists r.C, C \sqsubseteq D, \exists r.D \sqsubseteq D\}$$

Neither of the two contains any axioms that are entailed by the remainder of the ontology. There are also no sub-expressions that can be removed. However, $\mathcal{T}_2$ is less complex than $\mathcal{T}_1$, because $C \sqsubseteq D$ is simpler (shorter) than $\exists r.C \sqsubseteq \exists r.D$.

While the above problem is already known to be non-tractable and can have many solutions, the ability to rewrite axioms of the ontology can further increase the difficulty and the number of possible solutions: While in the above cases a minimal ontology contains only subconcepts $\mathtt{sub}(\mathcal{T})$ of the original ontology $\mathcal{T}$, in general, a minimal ontology can introduce new concept expressions as demonstrated in the following example.

Example 4. Consider the following TBox $\mathcal{T}$:

$$\begin{array}{cc} C_1 \sqsubseteq A_2 & A_2 \sqsubseteq C_3 \\ \exists r.D \sqsubseteq D & \exists s.C_1 \sqsubseteq D \end{array}$$

$$\exists s.C_3 \sqsubseteq \exists r.(\exists s.C_1)$$

Assume that $\int(C_1)$ and $\int(C_3)$ are large. Then the axiom $\exists s.C_1 \sqsubseteq D$ needs to be exchanged by $\exists s.A_2 \sqsubseteq D$ to obtain a smaller TBox. The TBox $\mathcal{T}_m$ given below is a minimal representation of $\mathcal{T}$.

$$\begin{array}{cc} C_1 \sqsubseteq A_2 & A_2 \sqsubseteq C_3 \\ \exists r.D \sqsubseteq D & \exists s.A_2 \sqsubseteq D \end{array}$$

$$\exists s.C_3 \sqsubseteq \exists r.(\exists s.C_1)$$

We notice that the original ontology $\mathcal{T}$ does not contain the expression $\exists s.A_2 \in \mathtt{sub}(\mathcal{T}_m)$.

We can conclude that considering subsumption relations between subconcepts $\mathtt{sub}(\mathcal{T})$ of $\mathcal{T}$ is not sufficient when looking for a minimal equivalent representation. In the next section, we show that the corresponding decision problem **P1** is in fact NP-complete.

4 NP-Completeness

In this section, we first show the NP-hardness of the problem and then establish its NP-completeness. We show NP-hardness by a reduction from the set cover problem, which is one of the standard NP-complete problems. For a given set $\mathcal{S} = \{S_1, S_2, \ldots, S_n\}$ with carrier set $S = \bigcup_{i=1}^{n} S_i$, a *cover* $\mathcal{C} \subseteq \mathcal{S}$ is a subset of $\mathcal{S}$, such that the union of the sets in $\mathcal{C}$ covers S, i.e., $S = \bigcup_{C \in \mathcal{C}} C$.

The *set cover problem* is the problem to determine, for a given set $\mathcal{S} = \{S_1, S_2, \ldots, S_n\}$ and a given integer k, if there is a cover $\mathcal{C}$ of $\mathcal{S}$ with at most $k \geq |\mathcal{C}|$ elements. We will use a restricted version of the set cover problem, which we call the *dense set cover problem* (DSCP). In the dense set cover problem, we require that

- neither the carrier set S nor the empty set is in $\mathcal{S}$,
- all singleton subsets (sets with exactly one element) of S are in $\mathcal{S}$, and
- if a non-singleton set S is in $\mathcal{S}$, so is some subset $S' \subseteq S$, which contains only one element less than S ($|S \smallsetminus S'| = 1$).

Lemma 1. *The dense set cover problem is NP-complete.*

Proof Sketch. For the full version of the proof, see extended version of the paper. The proof shows how to convert the cover of the non-dense set into a cover of the corresponding dense set and vise versa. □

Given the above NP-completeness result, we show that the size of minimal equivalents specified in **P1** is a linear function of the size of the minimal cover. To this end, we use the lemma below to obtain a lower bound on the size of equivalents. Intuitively, it states that for each entailed non-trivial equivalence $C \equiv A$, the TBox must contain at least one axiom that is at least as large as $C' \equiv A$ for some C' with $\mathcal{T} \models C \equiv C'$:

Lemma 2. *Let $\mathcal{T}$ be an $\mathcal{EL}$ TBox, $A \in sig(\mathcal{T})$ and C, D $\mathcal{EL}$ concepts such that $\mathcal{T} \models C \equiv A$, $\mathcal{T} \models A \sqsubseteq D$ (the latter is required for induction). Then, one of the following is true:*

1. *A is a conjunct of C (including the case $C = A$);*
2. *there exists an $\mathcal{EL}$ concept C' such that $\mathcal{T} \models C \equiv C'$ and $C' \bowtie A \in \mathcal{T}$ or $C' \bowtie A \sqcap D' \in \mathcal{T}$ for some $\bowtie \in \{\equiv, \sqsubseteq\}$ and some concept D'.*

Proof Sketch. For the full version of the proof, see extended version of the paper. We use the sound and complete proof system for general subsumption in $\mathcal{EL}$ terminologies introduced in [8] and prove the lemma by induction on the depth of the derivation of $C \sqsubseteq A \sqcap D$. We assume that the proof has minimal depth and consider the possible rules that could have been applied last to derive $C \sqsubseteq A \sqcap D$. In each case the lemma holds. □

We now show how to encode the dense set cover problem as an ontology minimization problem. Consider an instance of the dense set cover problem with the carrier set $A = \{B_1, \ldots, B_n\}$, the set $\mathcal{S} = \{A_1, \ldots, A_m, \{B_1\}, \ldots, \{B_n\}\}$ of subsets that can be used to form a cover. By interpreting the set and element names as atomic concepts, we can construct $\mathcal{T}_{\mathcal{S}\mathtt{base}}$ as follows:

$$\mathcal{T}_{\mathcal{S}\mathtt{base}} = \{A'' \equiv A' \sqcap B \mid A'', A' \in \mathcal{S}, B \in A, A'' = A' \cup \{B\}, A'' \neq A'\}.$$

Observe that the size of $\mathcal{T}_{\mathcal{S}\mathtt{base}}$ is at least $3m$. Clearly, $\mathcal{T}_{\mathcal{S}\mathtt{base}} \models A_i \equiv \bigsqcap_{B \in A_i} B$. Let $\mathcal{T}_{\mathcal{S}} = \mathcal{T}_{\mathcal{S}\mathtt{base}} \cup \{A \equiv \bigsqcap_{B \in A} B\}$. We establish the connection between the size of $\mathcal{T}_{\mathcal{S}}$ equivalents and the size of the cover of $\mathcal{S}$ as follows:

Lemma 3. *$\mathcal{T}_{\mathcal{S}}$ has an equivalent of size $\int(\mathcal{T}_{\mathcal{S}\mathtt{base}}) + k + 1$ if, and only if, $\mathcal{S}$ has a cover of size k.*

Proof. For the if-direction, assume that $\mathcal{S}$ has a cover of size k. We construct $\mathcal{T}'_{\mathcal{S}}$ of size $\int(\mathcal{T}_{\mathcal{S}\mathtt{base}}) + k + 1$ as follows: $\mathcal{T}'_{\mathcal{S}} = \mathcal{T}_{\mathcal{S}\mathtt{base}} \cup \{A \equiv \bigsqcap_{A' \in \mathcal{C}} A'\}$. Clearly, $\mathcal{T}'_{\mathcal{S}} \equiv \mathcal{T}_{\mathcal{S}}$.

For the only-if-direction, we assume that k is minimal and argue that no equivalent $\mathcal{T}' \in [\mathcal{T}_{\mathcal{S}}]$ of size $\leq \int(\mathcal{T}_{\mathcal{S}\mathtt{base}}) + k$ can exist. Assume that $\mathcal{T}$ is a minimal TBox with $\mathcal{T} \in [\mathcal{T}_{\mathcal{S}}]$. With the observation, that the $m+n$ atomic concepts that represent elements of $\mathcal{S}$ are pairwise not equivalent with each other or the concept A that represents the carrier set, we can conclude that no two atomic concepts are equivalent. From Lemma 2 it follows that, for each A_i with $i \in \{1, \ldots, m\}$, there is an axiom $C_i \equiv C'_i \in \mathcal{T}$ or $C_i \sqsubseteq C'_i \in \mathcal{T}$ such that $\mathcal{T} \models C_i \equiv A_i$ and A_i is a conjunct of C'_i or $A_i = C'_i$. Since there are no equivalent atomic concepts and $C_i \neq A_i$ due to the minimality of $\mathcal{T}$, the size of each such axiom is at least 3 and none of these axioms coincide. Additionally, since $\mathcal{T}_{\mathcal{S}} \not\models A_i \sqsubseteq A$, A cannot occur as a conjunct of C_i or as a conjunct of C'_i;

Finally, we estimate the size of the remaining axioms and show that their cumulative size is $> k$. It also follows from Lemma 2 that there exists an axiom $C \equiv C' \in \mathcal{T}$ or $C \sqsubseteq C' \in \mathcal{T}$ such that $\mathcal{T} \models C \equiv A$ and A is a conjunct of C' or $A = C'$. It holds that $\mathcal{T} \models C \equiv \bigsqcap_{B \in A} B$. We also know that for no proper subset $S' \subsetneq A$ holds $\mathcal{T} \models \bigsqcap_{B \in S'} B \sqsubseteq C$. Thus, we have found a cover of S and the size of the axiom must be $\geq k + 1$. Thus, the overall size of $\mathcal{T}$ must be $\geq \int(\mathcal{T}_{\mathcal{S}\mathtt{base}}) + k + 1$. □

Theorem 1. P1 *is in* NP.

Proof. We ask the non-deterministic algorithm to guess a TBox of the size $\leq k$. It remains to verify $\mathcal{T}' \equiv \mathcal{T}$, which can be done in PTIME [7]. □

Theorem 2. P1 *is* NP*-complete.*

Proof. The problem is NP-hard as an immediate consequence of Lemmas 3 and 1. Given the result of Theorem 1, we establish NP-completeness of the problem. □

5 Minimizing Acyclic TBoxes

In this section, we develop an algorithm for minimizing TBoxes in polynomial time, which is guaranteed to provide a minimal TBox for a class of $\mathcal{EL}$ TBoxes satisfying a certain type of acyclicity conditions. The algorithm can also be applied to more expressive and some cyclic TBoxes, however without the guarantee of minimality.

5.1 Acyclicity Conditions

In this subsection, we introduce equivalence classes on concepts and discuss cyclic dependencies between equivalence classes and their impact on computing minimal representations. Let $\mathcal{T}$ be an $\mathcal{EL}$ TBox and let C be a concept in $\mathtt{sub}(\mathcal{T})$. We use the notation $[C]_{\mathcal{T}} = \{C' \in \mathtt{sub}(\mathcal{T}) \mid \mathcal{T} \models C \equiv C'\}$ to denote the *equivalence class* of the concept C and $\mathcal{C}_{\mathcal{T}} = \{[C]_{\mathcal{T}} \mid C \in \mathtt{sub}(\mathcal{T})\}$ to denote the set of all equivalence classes over the set $\mathtt{sub}(\mathcal{T})$. In case $\mathcal{T}$ is clear from the context, we omit the index. We base the acyclicity conditions on the following reference relations, which use both syntactic and semantic dependencies between equivalence classes:

Definition 3. *Let $\mathcal{T}$ be an $\mathcal{EL}$ TBox. The* reference relations $\prec_{\sqsubseteq}, \prec_{\sqsupseteq}$ *and* $\prec_s$*, all subsets of $\mathcal{C} \times \mathcal{C}$, are given as follows:*

- $[C] \prec_s [C']$ *if, for some $C_1 \in [C], C_2 \in [C']$, it holds that C_2 occurs in C_1;*
- $[C] \prec_{\sqsubseteq} [C']$ *if, for some $C_1 \in [C], C_2 \in [C']$, it holds that $[C_1] \prec_s [C_2]$ or $\mathcal{T} \models C_1 \sqsubseteq C_2$;*
- $[C] \prec_{\sqsupseteq} [C']$ *if, for some $C_1 \in [C], C_2 \in [C']$, it holds that $[C_1] \prec_s [C_2]$ or $\mathcal{T} \models C_1 \sqsupseteq C_2$.*

We call a TBox *cyclic*, if any of the above relations $\prec_{\sqsubseteq}, \prec_{\sqsupseteq}, \prec_s$ is cyclic. We say that a TBox $\mathcal{T}$ is *strongly cyclic* if $\prec_s$ is cyclic. The algorithm presented in this paper is applicable for TBoxes not containing strong cycles. Most of the large bio-medical ontologies including Galen, Gene Ontology and NCI do not contain strong cycles. This was also the case for earlier versions of SNOMED, e.g., the one dated 09 February 2005 [9]. Note that asking for the absence of cycles in $\prec_s$ is a weaker requirement than for $\prec_{\sqsubseteq}$ or $\prec_{\sqsupseteq}$, as $\prec_s \subseteq \prec_{\sqsubseteq} \cap \prec_{\sqsupseteq}$. But the reverse relationship between the conditions holds.

In some cases, TBoxes contain cycles that are caused by redundant conjuncts and can easily be removed.

Example 5. $\{A \sqcap B \sqsubseteq C, A \sqsubseteq B\}$ has a cyclic $\prec_{\sqsupseteq}$ relation due to a cycle between $A \sqcap B$ and A. It can be transformed into an acyclic TBox $\{A \sqsubseteq C, A \sqsubseteq B\}$.

We call conjunctions $C' \sqcap C''$ in $\mathtt{sub}(\mathcal{T})$ such that $\mathcal{T} \models C' \sqsubseteq C''$ *subsumer-containing conjunctions*. We can easily eliminate subsumer-containing conjunctions in TBoxes before applying the algorithm: for each subsumer-containing conjunction $C' \sqcap C''$ in $\mathtt{sub}(\mathcal{T})$ with $\mathcal{T} \models C' \sqsubseteq C''$, we replace $C' \sqcap C''$ in $\mathcal{T}$ by C', and add the axiom $C' \sqsubseteq C''$ to $\mathcal{T}$. We can show that the closure of each equivalence class $[C]$ of an acyclic TBox $\mathcal{T}$ is finite if we exclude subsumer-containing conjunctions. We denote such a closure with $[C]^* = \{C' \mid \mathcal{T} \models C \equiv C'$ and C' is not a subsumer-containing conjunction$\}$. We denote the extended set of subconcepts of $\mathcal{T}$ by $\mathtt{sub}(\mathcal{T})^* = \bigcup_{[C] \in \mathcal{C}} [C]^*$.

Another kind of removable cyclic dependencies are conjunctions on the right-hand side. We use a simple *decomposition*, in which all conjunctions on the right-hand side of axioms are replaced by separate inclusion axioms for each conjunct. We obtain the decomposed version $\mathcal{T}'$ of a TBox $\mathcal{T}$ by replacing each $C \sqsubseteq D_1 \sqcap D_2 \in \mathcal{T}_m$ by $C \sqsubseteq D_1, C \sqsubseteq D_2$ until a fixpoint is reached. *Composition* is the dual transformation:

we replace any two axioms $C \sqsubseteq D_1, C \sqsubseteq D_2$ by $C \sqsubseteq D_1 \sqcap D_2$ until a fixpoint is reached.

Unless we state otherwise, in the following we assume that TBoxes are decomposed and do not contain subsumer-containing conjunctions.

5.2 Uniqueness of Minimal TBoxes

Acyclic TBoxes are better behaved not only with respect to the complexity of minimization, but they also have a unique minimal TBox modulo replacement of equivalent concepts by one another (if we assume that the TBox with the lower number of equivalence axioms should be preferred in case of equally large TBoxes).

To be able to determine a unique syntactic representation of a TBox $\mathcal{T}$, we choose a representative $C' \in [C]^*$ for each equivalence class $[C] \in \mathcal{C}$ and denote it using the *representative selection function* $r : \mathcal{C} \to \mathtt{sub}(\mathcal{T})^*$ with $r([C]) = C'$. We say that r is *valid*, if for all $[C], [D] \in \mathcal{C}$ with $[C] \neq [D]$ it holds that $C' \in [C]^*$ occurs in $r([D])$ only if $C' = r([C])$, i.e., representatives can only contain other representatives, but not other elements of equivalence classes.

Definition 4. *Let $\mathcal{T}$ be a TBox and $\bowtie \in \{\equiv, \sqsubseteq\}$. We say that $\mathcal{T}$ is* aligned *with r, if for each $C \bowtie D \in \mathcal{T}$ one of the following conditions holds:*

- *if $\mathcal{T} \not\models C \equiv D$, then $C = r([C])$ and $D = r([D])$;*
- *if $\mathcal{T} \models C \equiv D$, then for each C' such that $C' \neq C$, $C' \neq D$ and C' occurs in C or D it holds that $C' = r([C'])$.*

In other words, the only axioms, in which we allow an occurrence of a non-representative C are axioms relating C with concepts equivalent to it.

Since minimal TBoxes can sometimes contain subsumption axioms relating two equivalent concepts with each other, the otherwise unique TBox result can vary in the choice between subsumption and equivalence axioms. For the sake of uniqueness, we assume that, whenever we have a choice between equivalence ($\equiv$) and subsumption axioms ($\sqsubseteq$) in the resulting TBox, we prefer subsumption axioms.

We call a TBox *non-redundant*, if there is no $\alpha \in \mathcal{T}$ such that $\mathcal{T} \setminus \{\alpha\} \models \alpha$. In order to show how to compute a minimal equivalent TBox for an acyclic initial TBox, we first show that we do not need new equivalence classes or new relations between them to obtain any non-redundant, decomposed, equivalent TBox. In other words, non-redundant, decomposed axioms encoding relations between equivalence classes are unique up to exchanging equivalent concepts.

Lemma 4. *Let $\mathcal{T}_1, \mathcal{T}_2$ be two non-redundant, acyclic $\mathcal{EL}$ TBoxes such that $\mathcal{T}_1 \equiv \mathcal{T}_2$. Let $C \sqsubseteq D \in \mathcal{T}_2$. Then there is $C' \sqsubseteq D' \in \mathcal{T}_1$ such that $\mathcal{T}_1 \models C' \equiv C$, $\mathcal{T}_1 \models D' \equiv D$.*

While the above lemma addresses relations between equivalence classes in non-redundant, decomposed TBoxes, it does not allow us to draw conclusions about axioms representing relations within equivalence classes. The purpose of the below lemma is to determine the part of the TBox that encodes relations between equivalent concepts within equivalence classes. For this, we divide the TBox into partitions: one for non-equivalence axioms $\mathcal{T}^0 = \{C \sqsubseteq D \in \mathcal{T} \mid \mathcal{T} \not\models C \equiv D\}$ and one for axioms encoding

relations within each equivalence class: $\mathcal{T}^{[C']} = \{C \equiv D \in \mathcal{T} \mid C, D \in [C']\}$ for each $[C'] \in \mathcal{C}$. We denote the set of all subsumption dependencies holding within a partition by $\mathcal{T}^{\mathtt{full},[C']} = \{C \sqsubseteq D \mid C, D \in [C']\}$. In each (equivalence class) partition, a part of dependencies can be deducible from the remainder of the TBox.

Example 6. Consider the TBox $\mathcal{T} = \{A \sqsubseteq B, \exists r.A \equiv \exists r.B\}$. For the equivalence class $\{\exists r.A, \exists r.B\}$, the subsumption $\exists r.A \sqsubseteq \exists r.B$ follows from $A \sqsubseteq B$.

We denote entailed dependencies for an equivalence class $[C']$ by $\mathcal{T}^{\mathtt{red},[C']} = \{C \sqsubseteq D \mid C, D \in [C']\}$. We now consider alternative representations of each partition $\mathcal{T}^{[C']}$. We first show that, in any acyclic TBox $\mathcal{T}$ aligned with some valid r, we can determine the entailed dependencies $\mathcal{T}^{\mathtt{red},[C']}$ within each $\mathcal{T}^{\mathtt{full},[C']}$ based on $\mathcal{T}^0$.

Lemma 5. *Let $\mathcal{T}$ be a non-redundant, acyclic $\mathcal{EL}$ TBox aligned with a valid representative selection function r. Then, for each non-singleton equivalence class $[C'] \in \mathcal{C}(\mathcal{T})$ and each pair $C, D \in [C']$, it holds that $C \sqsubseteq D \in \mathcal{T}^{\mathtt{red},[C']}$ exactly if one of the following conditions is true:*

1. $D = \top$
2. *there are concepts C', D' such that $C = \exists r.C', D = \exists r.D'$ and $\mathcal{T} \models C' \sqsubseteq D'$, $\mathcal{T} \not\models C' \equiv D'$.*

As a consequence, each equivalence class partition can be considered independently from other equivalence class partitions. In particular, this implies that, for any syntactic representation $\mathcal{T}^{[C]}$ of a partition for equivalence class $[C']$, we can obtain $\mathcal{T}^{\mathtt{full},[C']}$ from $\mathcal{T}^{[C]} \cup \mathcal{T}^{\mathtt{red},[C]}$ by computing its transitive closure [2].

Lemma 6. *Let $\mathcal{T}$ be a non-redundant, acyclic $\mathcal{EL}$ TBox aligned with a valid representative selection function r. Then, for each equivalence class $[C] \in \mathcal{C}(\mathcal{T})$ it holds that $(\mathcal{T}^{[C]} \cup \mathcal{T}^{\mathtt{red},[C]})^* = \mathcal{T}^{\mathtt{full},[C']}$.*

Since our implementation operates on ontologies represented in the OWL Web Ontology Language, we consider here an important detail of this language. In addition to constructs mentioned in preliminaries, OWL Web Ontology Language allows for *OwlEquivalentClassesAxioms* - axioms, in which we can specify a set of equivalent concepts. With the exception of equivalence classes containing $\top$, for which there exists an equally small representation without an OwlEquivalentClassesAxiom, this is clearly the smallest representation for equivalence class partitions.

Let $[C]^{\mathtt{nonred}} = [C] \setminus \{C' \in [C] \mid C' \sqsubseteq D'_1$ and $C' \sqsupseteq D'_2 \in \mathcal{T}^{\mathtt{red},[C]}$ for some $D'_1, D'_2\}$. Let $\mathcal{T}^{\mathtt{nonred},[C]}$ be the corresponding OWLEquivalentClassesAxiom with $[C]^{\mathtt{nonred}}$ as the set of equivalent concepts. Note that, according to the semantics of OwlEquivalentClassesAxioms, it holds that $\mathcal{T}^{\mathtt{nonred},[C]} \models \mathcal{T}^{\mathtt{full},[C]^{\mathtt{nonred}}}$. Thus, $\mathcal{T}^{\mathtt{nonred},[C]} \cup \mathcal{T}^{\mathtt{red},[C]} \models \mathcal{T}^{\mathtt{full},[C]}$. Note that $\int(\mathcal{T}^{\mathtt{nonred},[C]}) = \sum_{C' \in [C]^{\mathtt{nonred}}} \int(C')$.

Lemma 7. *Let $\mathcal{T}$ be a non-redundant, acyclic $\mathcal{EL}$ TBox aligned with a valid representative selection function r. Then, $\int(\mathcal{T}^{\mathtt{nonred},[C]}) \leq \int(\mathcal{T}^{[C]})$ for each equivalence class $[C] \in \mathcal{C}(\mathcal{T})$.*

[2] For a set $\mathcal{T}$ of axioms, the transitive closure $(\mathcal{T})^*$ is obtained by including $C \sqsubseteq D$ for any C, D such that there exists C' with $\mathcal{T} \models \{C \sqsubseteq C', C' \sqsubseteq D\}$.

Algorithm 1. Rewriting $\mathcal{T}_{\mathtt{in}}$

Data: $\mathcal{T}_{\mathtt{in}}$ acyclic decomposed TBox
Result: $\mathcal{T}_{\mathtt{out}}$ minimal equivalent TBox

1 $\mathcal{C}_{\mathtt{all}} \leftarrow \mathcal{C}$;
2 $\mathcal{C}_{\mathtt{TODO}} \leftarrow \mathcal{C}_{\mathtt{all}}$;
3 $\mathcal{T}_{\mathtt{out}} \leftarrow$ remove equivalence axioms from $\mathcal{T}_{\mathtt{in}}$;
4 **while** $\mathcal{C}_{\mathtt{TODO}} \neq \emptyset$ **do**
5 **for** $[C] \in \mathit{leaves}(\mathcal{C}_{\mathtt{TODO}})$ **do**
6 choose minimal representative $r([C])$;
7 replace $C' \in [C]$ in $\mathcal{T}_{\mathtt{out}}$ by $r([C])$;
8 replace $C' \in [C]$ in $\mathcal{C}_{\mathtt{TODO}} \setminus \{[C]\}$ by $r([C])$;
9 replace $C' \in [C]$ in $\mathcal{C}_{\mathtt{all}} \setminus \{[C]\}$ by $r([C])$;
10 $\mathcal{C}_{\mathtt{TODO}} \leftarrow \mathcal{C}_{\mathtt{TODO}} \setminus \{[C]\}$;
11 $\mathcal{T}_e \leftarrow \bigcup_{[C] \in \mathcal{C}_{\mathtt{all}}, |[C]| \geq 2} \mathcal{T}^{\mathtt{nonred},[C]}$;
12 **for** $\alpha \in \mathcal{T}_{\mathtt{out}}$ **do**
13 **if** $\mathcal{T}_{\mathtt{out}} \cup \mathcal{T}_e \setminus \{\alpha\} \models \alpha$ **then**
14 $\mathcal{T}_{\mathtt{out}} \leftarrow \mathcal{T}_{\mathtt{out}} \setminus \{\alpha\}$;
15 $\mathcal{T}_{\mathtt{out}} \leftarrow \mathcal{T}_{\mathtt{out}} \cup \mathcal{T}_e$;
16 $\mathcal{T}_{\mathtt{out}} \leftarrow \mathrm{compose}(\mathcal{T}_{\mathtt{out}})$;

Based on the above two lemmas, we can show that, in the acyclic case, we can compute a minimal TBox by eliminating redundant axioms, fixing the representative selection function r to some minimal value, constructing the core representation $\mathcal{T}^{\mathtt{nonred},[C]}$ for each non-singleton equivalence class $[C]$ and composing $\mathcal{T}$ again.

Definition 5. *Let $\mathcal{T}$ be an $\mathcal{EL}$ TBox and r a corresponding valid representative selection function. We say that r is* minimal*, if for each $[C] \in \mathcal{C}$ holds: there is no $C \in [C]^*$ such that $\int(C) < \int(r([C]))$.*

We can now state the minimality of the composed TBox containing $\mathcal{T}^0$ and a partition $\mathcal{T}^{\mathtt{nonred},[C]}$ for each non-singleton equivalence class $[C] \in \mathcal{C}$.

Theorem 3. *Let $\mathcal{T}$ be a non-redundant, acyclic $\mathcal{EL}$ TBox and r a minimal, valid representative selection function. Let the TBox $\mathcal{T}_n = \mathcal{T}^0 \cup \bigcup_{[C] \in \mathcal{C}, |[C]| \geq 2} \mathcal{T}^{\mathtt{nonred},[C]}$ be aligned with r. Let $\mathcal{T}'_n$ be a composed version of $\mathcal{T}_n$. Then, for any minimal TBox $\mathcal{T}_m$ with $\mathcal{T}_m \equiv \mathcal{T}$ it holds that $\int(\mathcal{T}_m) = \int(\mathcal{T}'_n)$.*

Algorithm 1 implements the iterative computation of r and the minimal TBox $\mathcal{T}'_n$. It takes an acyclic decomposed TBox $\mathcal{T}_{\mathtt{in}}$ and computes the corresponding minimal equivalent TBox $\mathcal{T}_{\mathtt{out}}$. Line 3 is not strictly necessary, but allows for a more efficient processing. In Lines 4-10, a minimal representative selection function r is iteratively determined – for one equivalence class at a time – and all data structures are aligned with r. We distinguish two versions of equivalence classes: $\mathcal{C}_{\mathtt{TODO}}$ contains equivalence classes, for which the minimal representative has not been selected yet. In each iteration, we process the leaves in $\mathcal{C}_{\mathtt{TODO}}$ ordered with the reference relation $\prec_s$ and remove those

equivalence classes from $\mathcal{C}_{\texttt{TODO}}$. $\mathcal{C}_{\texttt{all}}$ contains all equivalence classes that are stepwise aligned with a minimal representative selection function r. In each step, we also align axioms $\mathcal{T}_{\texttt{out}}$ corresponding to the partition $\mathcal{T}^0$ with r by replacing concepts with the representative $r([C])$ fixed in Line 6.

In Line 11, we build partitions for non-singleton equivalence classes. In Lines 12-14, we compute the non-redundant part of $\mathcal{T}_{\texttt{out}}$. The function $\texttt{compose}(\mathcal{T}_{\texttt{out}})$ in Line 16 composes subsumption axioms with identical left-hand sides into a single axiom.

Clearly, Algorithm 1 runs in PTIME. $\texttt{sub}(\mathcal{T})$ is polynomially large in the size of $\mathcal{T}$ and $\mathcal{C}$ can be computed in PTIME due to tractable reasoning in $\mathcal{EL}$. Equivalence axioms can be removed in linear time. Lines 4-10 are executed $|\mathcal{C}|$ times and can be executed in PTIME. The same holds for building partitions for non-singleton equivalence classes and computing the non-redundant part of $\mathcal{T}_{\texttt{out}}$. Composition can be performed in linear time. Note that the algorithm remains tractable only assuming the tractability of reasoning in the underlying logic. Otherwise, the complexity of reasoning dominates. In principle, the result could be obtained after computing the representatives for each equivalence class by simply selecting all subsumption relations between classes. However, this would result in a less efficient implementation with large intermediary results.

Theorem 4. *Let $\mathcal{T}$ be an acyclic $\mathcal{EL}$ TBox. Algorithm 1 computes a minimal equivalent TBox in* PTIME.

Minimality is a consequence of Theorem 3. Equivalence follows from $\mathcal{T}^{\texttt{nonred},[C]} \cup \mathcal{T}^{\texttt{red},[C]} \models \mathcal{T}^{\texttt{full},[C]}$ for each non-singleton equivalence class $[C]$ and from Lemma 4.

6 Experimental Results

For our evaluation, we have implemented the algorithm using the latest version of OWL API and Hermit reasoner. We have used an optimized version of Algorithm 1, where entailment checking is done in two phases, the first of which can be run by several threads.

A selection of publicly available ontologies (as shown in Table 1) that vary in size and expressivity have been used in the experiments[3]. Table 2 shows the number $|\texttt{CON}_o(\mathcal{T})|$ of occurrences of complex concepts $\texttt{CON}(\mathcal{T}) = \texttt{sub}(\mathcal{T}) \setminus \text{sig}_C(\mathcal{T})$ in the first two columns (the original value followed by the new value relative to the original one). The two subsequent columns show the number of pairwise different complex concepts $|\texttt{CON}(\mathcal{T})|$. The last two columns show $\int(\mathcal{T})$ – the size of each ontology measured as the number of occurrences of entities in $\text{sig}(\mathcal{T})$.

The implementation was first applied to Snomed [10]. However, the available fully-fledged reasoners Pellet and Hermit run out of heap space when classifying the ontology even with 10 GB memory assigned to the corresponding Java process. The ELK reasoner [11] is capable of classifying Snomed, but it does not currently implement entailment, which is essential for our implementation.

[3] The wine ontology can be retrieved from `http://www.w3.org/TR/2003/PR-owl-guide-20031209/wine`. All other ontologies used can be found in the TONES ontology repository at `http://owl.cs.manchester.ac.uk/repository`

Table 1. Properties of ontologies used in experiments

	$\lvert\mathcal{T}\rvert$	$\int(\mathcal{T})/\lvert\mathcal{T}\rvert$	$\mathtt{CON}(\mathcal{T})/\lvert\mathcal{T}\rvert$	$\mathtt{CON}_o(\mathcal{T})/\lvert\mathcal{T}\rvert$	Logic
Snomed	83,259	4.99	1.14	2.57	$\mathcal{EL}++$
Gene Ontology	42656	3.37	1.20	0.27	$\mathcal{EL}++$
NCI	97811	1.10	0.00	0.14	$\mathcal{ALCH}(\mathcal{D})$
Galen	4735	2.81	0.52	1.13	$\mathcal{ALEHIF}+$
Adult Mouse	3464	2.48	0.15	0.48	$\mathcal{EL}++$
Wine	657	1.03	0.21	0.40	$\mathcal{SHOIN}(\mathcal{D})$
Nautilus	38	2.18	0.29	0.40	$\mathcal{ALCHF}(\mathcal{D})$
Cell	1264	2.16	0.09	0.16	$\mathcal{EL}++$
DOLCE-lite	351	1.42	0.13	0.14	$\mathcal{SHIF}$
Software	238	25.21	2.60	7.26	$\mathcal{ALHN}(\mathcal{D})$
Family Tree	36	6.19	1.02	1.33	$\mathcal{SHIN}(\mathcal{D})$
General Ontology	8803	0.48	0.03	0.03	$\mathcal{ALCHOIN}(\mathcal{D})$
Substance	609	2.33	0.22	0.36	$\mathcal{ALCHO}(\mathcal{D})$
Generations	38	1.87	0.58	1.21	$\mathcal{ALCOIF}$
Periodic Table	58	1.38	0.38	0.43	$\mathcal{ALU}$

From the ontologies used in our experiments, only Snomed did not satisfy the acyclicity conditions for $\prec_s$ sufficient to guarantee termination of our algorithm. On the one hand, Snomed contains cyclic concept definitions. For instance, `Mast_cell_leukemia` is defined by means of the corresponding equivalence axiom as

$$\begin{aligned}&\mathtt{Leukemia\_disease} \sqcap\\ &\mathtt{Mast\_cell\_malignancy} \sqcap\\ &\exists\,\mathtt{RoleGroup}.\\ &\quad(\exists\,\mathtt{Associated\_morphology}.\mathtt{Mast\_cell\_leukemia} \sqcap\\ &\quad\exists\,\mathtt{Finding\_site}.\mathtt{Hematopoietic\_system\_structure})) \sqcap\\ &\exists\,\mathtt{RoleGroup}.(\\ &\quad\exists\,\mathtt{Has\_definitional\_manifestation}.\mathtt{White\_blood\_cell\_finding})\end{aligned}$$

On the other hand, Snomed contains a cyclic reference relation between the concepts `Wound` and `Wound_finding`, which is the only cyclic dependency with more than one element.

We have manually evaluated how the rewriting has affected ontologies. In all cases where concepts became smaller, the improvement has been achieved by either elimination of redundant axioms or exchanging complex expressions by atomic concepts.

In case of the Galen ontology [1], the algorithm managed to reduce the number of occurrences of complex concepts by 955, which is 17%. The size of the ontology in number of references was reduced by 1130 (9%). The number of distinct complex concepts used in the ontology was reduced by 76 (3%). The situation is similar for the NCI [12] ontology.

The other large medical ontology – Gene Ontology [13] – does not contain any equivalent concepts, i.e., each equivalence class has only one element. The current algorithm did not find any possibility to rewrite the ontology. The same holds for Adult Mouse and Periodic Table ontologies.

Table 2. Minimization results for different ontologies

	$\mathtt{CON}_o(\mathcal{T})$		$\lvert\mathtt{CON}(\mathcal{T})\rvert$		$\int(\mathcal{T})$	
Snomed	213,856	–	95,315	–	415,541	–
Gene Ontology	11,686	1	8,508	1	143,900	1
NCI	13,961	0.87	4,000	0.99	107,841	0.94
Galen	5,368	0.83	2,475	0.97	13,285	0.91
Adult Mouse	1,649	0.99	507	1	8,575	0.99
Wine	262	0.89	141	0.98	677	0.93
Nautilus	15	1	11	1	83	0.86
Cell	206	0.87	114	0.96	2,732	0.96
DOLCE-lite	49	0.92	46	0.98	497	0.66
Software	1,728	0.81	620	1	6,001	0.81
Family Tree	48	0.77	37	0.78	223	0.83
General Ontology	281	0.83	278	0.83	4,182	0.83
Substance	221	1	135	1	1,417	0.95
Generations	46	0.65	22	1	71	0.90
Periodic Table	25	1	22	1	80	1

Results for the other, relatively small ontologies are similar to those of Galen and in some cases more prominent (Table 2). The highest improvement (66% of $\int(\mathcal{T})$) was achieved in the DOLCE-Lite ontology [14].

7 Related Work

The work on knowledge compilation [15] is closely related to the work presented in this paper. Knowledge compilation is a family of approaches, in which a knowledge base is transformed in an off-line phase into a normal form, for which reasoning is cheaper. The hope is that the one-off cost of the initial preprocessing will be justified by the computational savings made on subsequent reasoning. One of such normal forms proposed in description logics is the prime implicates normal form for $\mathcal{ALC}$ ontologies [4]. Prime implicates of a logical formula are defined to be their strongest clausal consequences. Concepts in the prime implicates normal form are expected to be easier to read and understand. Reasoning is also expected to be more efficient for knowledge bases in this normal form. For example, concept subsumption can be tested in quadratic time. However, the problem with such normal forms is the blowup caused by the transformation. For $\mathcal{ALC}$ ontologies, a doubly-exponential blowup in the concept size can occur. Given that reasoning in $\mathcal{ALC}$ is PSPACE-complete [16], such a transformation can be disadvantageous in general.

Grimm et al. [3] propose two different algorithms for eliminating semantically redundant axioms from ontologies, which is one of the sources of non-succinctness. However, as shown in Section 3, it does not guarantee that we obtain a minimal TBox in $[(]\mathcal{T})$. The advantage of this restricted approach to improving succinctness is that the result contains only axioms that are familiar to the users of the ontology.

Work on laconic and precise justifications [17] (minimal parts of the ontology implying a particular axiom or axioms) is also related to this paper. The authors propose

an algorithm for computing laconic justifications – justifications that do not contain any logically superfluous parts. Laconic justifications can then be used to derive precise justifications – justifications that consist of flat, small axioms, and are important for the generation of semantically minimal repairs.

Nikitina et al. [18] propose an algorithm for an efficient handling of redundancy in inconsistent ontologies during their repair. Similarly to the approach by Grimm et al. axioms are considered as atoms that cannot be further separated into parts.

8 Summary and Outlook

We have considered the problem of finding minimal equivalent representations for ontologies expressed in the lightweight description logic $\mathcal{EL}$. We have shown that the task of finding such a representation (or rather: its related decision problem) is NP-complete. Further, we have identified a class of TBoxes for which the problem is tractable. We have implemented a polynomial algorithm for minimizing the above class of TBoxes. For general TBoxes, the algorithm can be used as a heuristic. We have implemented the algorithm and presented experimental results, which show that the complexity of various existing ontologies can be improved. For instance, in case of Galen, the number of complex concepts occurrences could be reduced by 955 and the number of references to atomic concepts and roles by 1130.

There are various natural extensions of this work. Inspired by recent results on uniform interpolation in $\mathcal{EL}$ [8], the problem can be extended to finding minimal representations for ontologies using a signature extension. The results in [8] imply that, even for the minimal equivalent representation of an ontology, an up to triple-exponentially more succinct representation can be obtained by extending its signature. Auxiliary concept symbols are therefore important contributors towards the succinctness of ontologies, e.g., used as shortcuts for complex $\mathcal{EL}$ concepts or disjunctions thereof. The results of our evaluation indicate that there are many complex concept expression that occur repeatedly in ontologies but do not have an equivalent atomic concept that could be used instead. Therefore, introducing names for such frequently used concepts could yield a further decrease of the ontology's complexity.

The results obtained within this paper can be transferred to the context of ontology reuse, where rewriting is applied to obtain a compact representation of the facts about a subset of terms [19], in particular in its extended form as suggested above.

Finally, minimizing representations is an interesting problem for knowledge representation formalisms in general, and similar questions can (and should) be asked for more expressive ontology languages.

References

1. Rector, A., Gangemi, A., Galeazzi, E., Glowinski, A., Rossi-Mori, A.: The GALEN CORE model schemata for anatomy: Towards a re-usable application-independent model of medical concepts. In: Proceedings of the 12th International Congress of the European Federation for Medical Informatics (MIE 1994), pp. 229–233 (1994)
2. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P.: The Description Logic Handbook: Theory, Implementation and Applications. Cambridge University Press (2003)

3. Grimm, S., Wissmann, J.: Elimination of redundancy in ontologies. In: Antoniou, G., Grobelnik, M., Simperl, E., Parsia, B., Plexousakis, D., De Leenheer, P., Pan, J. (eds.) ESWC 2011, Part I. LNCS, vol. 6643, pp. 260–274. Springer, Heidelberg (2011)
4. Bienvenu, M.: Prime implicates and prime implicants: From propositional to modal logic. Journal of Artificial Intelligence Research (JAIR) 36, 71–128 (2009)
5. Motik, B., Cuenca Grau, B., Horrocks, I., Wu, Z., Fokoue, A., Lutz, C. (eds.): OWL 2 Web Ontology Language: Profiles. W3C Recommendation (October 27, 2009), http://www.w3.org/TR/owl2-profiles/
6. OWL Working Group, W.: OWL 2 Web Ontology Language: Document Overview. W3C Recommendation (October 27, 2009), http://www.w3.org/TR/owl2-overview/
7. Baader, F., Brandt, S., Lutz, C.: Pushing the $\mathcal{EL}$ envelope. In: Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI 2005), pp. 364–369 (2005)
8. Nikitina, N., Rudolph, S.: ExpExpExplosion: Uniform interpolation in general EL terminologies. In: Proceedings of the 20th European Conference on Artificial Intelligence (ECAI 2012), pp. 618–623 (2012) (Shortlisted for best paper awards)
9. Konev, B., Walther, D., Wolter, F.: Forgetting and uniform interpolation in large-scale description logic terminologies. In: Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI 2009), pp. 830–835 (2009)
10. Spackman, K.A., Campbell, K.E., Cote, R.A.: Snomed rt: A reference terminology for health care. In: Proceedings of the AIMA Fall Symposium, pp. 640–644 (1997)
11. Kazakov, Y., Krötzsch, M., Simančík, F.: ELK reasoner: Architecture and evaluation. In: Proceedings of the OWL Reasoner Evaluation Workshop 2012 (ORE 2012). CEUR Workshop Proceedings, vol. 858. CEUR-WS.org (2012)
12. Sioutos, N., Coronado, S.D., Haber, M.W., Hartel, F.W., Shaiu, W.L., Wright, L.W.: Nci thesaurus: A semantic model integrating cancer-related clinical and molecular information. Journal of Biomedical Informatics 40(1), 30–43 (2007)
13. Ashburner, M.: Gene ontology: Tool for the unification of biology. Nature Genetics 25, 25–29 (2000)
14. Gangemi, A., Guarino, N., Masolo, C., Oltramari, A., Schneider, L.: Sweetening ontologies with dolce. In: Gómez-Pérez, A., Benjamins, V.R. (eds.) EKAW 2002. LNCS (LNAI), vol. 2473, p. 166. Springer, Heidelberg (2002)
15. Darwiche, A., Marquis, P.: A knowledge compilation map. Journal of Artificial Intelligence Research (JAIR) 17, 229–264 (2002)
16. Schmidt-Schauß, M., Smolka, G.: Attributive concept descriptions with complements. Artificial Intelligence 48(1), 1–26 (1991)
17. Horridge, M., Parsia, B., Sattler, U.: Laconic and precise justifications in owl. In: Sheth, A.P., Staab, S., Dean, M., Paolucci, M., Maynard, D., Finin, T., Thirunarayan, K. (eds.) ISWC 2008. LNCS, vol. 5318, pp. 323–338. Springer, Heidelberg (2008)
18. Nikitina, N., Rudolph, S., Glimm, B.: Reasoning-supported interactive revision of knowledge bases. In: Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI 2011), pp. 1027–1032 (2011)
19. Nikitina, N., Glimm, B.: Hitting the sweetspot: Economic rewriting of knowledge bases. In: Cudré-Mauroux, P., Heflin, J., Sirin, E., Tudorache, T., Euzenat, J., Hauswirth, M., Parreira, J.X., Hendler, J., Schreiber, G., Bernstein, A., Blomqvist, E. (eds.) ISWC 2012, Part I. LNCS, vol. 7649, pp. 394–409. Springer, Heidelberg (2012)

FedSearch: Efficiently Combining Structured Queries and Full-Text Search in a SPARQL Federation

Andriy Nikolov, Andreas Schwarte, and Christian Hütter

Fluid Operations AG, Walldorf, Germany
{andriy.nikolov,andreas.schwarte,christian.huetter}@fluidops.com

Abstract. Combining structured queries with full-text search provides a powerful means to access distributed linked data. However, executing hybrid search queries in a federation of multiple data sources presents a number of challenges due to data source heterogeneity and lack of statistical data about keyword selectivity. To address these challenges, we present FedSearch – a novel hybrid query engine based on the SPARQL federation framework FedX. We extend the SPARQL algebra to incorporate keyword search clauses as first-class citizens and apply novel optimization techniques to improve the query processing efficiency while maintaining a meaningful ranking of results. By performing on-the-fly adaptation of the query execution plan and intelligent grouping of query clauses, we are able to reduce significantly the communication costs making our approach suitable for top-k hybrid search across multiple data sources. In experiments we demonstrate that our optimization techniques can lead to a substantial performance improvement, reducing the execution time of hybrid queries by more than an order of magnitude.

1 Introduction

With the growing amount of Linked Data sources becoming available on the Web, full-text keyword search is becoming more and more important as a paradigm for accessing Linked Data. Already today the majority of triple stores support both full-text search and structured SPARQL queries, allowing for hybrid queries that combine these approaches. Given the distributed nature of Linked Data, efficient processing of user queries in a federated environment with multiple data sources has become a central research area in the Semantic Web Community [1,2].

In practice there are many use cases where hybrid search is required. Consider as an example a scenario involving a text-based database (e.g., a Semantic Wiki) that offers access to its data via a SPARQL interface (e.g., through LuceneSail). In addition, there might be one or more external RDF databases required to fulfill the information needs of the user.

However, execution of hybrid search queries presents several challenges at different levels. The first class of problems is caused by *data source heterogeneity*: Because there is no formal representation of full-text index search included in

H. Alani et al. (Eds.): ISWC 2013, Part I, LNCS 8218, pp. 427–443, 2013.

Table 1. Hybrid Search Queries for Different Selected Triple Stores

a) OWLIM[2]

```
SELECT ?page WHERE {
  ?id rdfs:label ?val .
  ?val luc:luceneIndex "obama" .
  ?nytId owl:sameAs ?id .
  ?nytId nyt:topicPage ?page .
}
```

b) Virtuoso[3]

```
SELECT ?page WHERE {
  ?id rdfs:label ?val .
. ?val bif:contains "obama" .
  ?nytId owl:sameAs ?id .
  ?nytId nyt:topicPage ?page .
}
```

c) LuceneSail[4]

```
SELECT ?page WHERE {
  ?id search:matches ?m .
  ?m search:query "obama" .
  ?m search:property rdfs:label .
  ?nytId owl:sameAs ?id .
  ?nytId nyt:topicPage ?page .
}
```

Triple store vendors use custom vocabularies to express keyword search: OWLIM uses the *http://www.ontotext.com/owlim/lucene#* namespace (luc), Vituoso uses a predefined *bif* prefix and LuceneSail uses the *http://www.openrdf.org/contrib/lucenesail#* namespace (search).

the standard SPARQL syntax[1], triple store manufacturers model keyword search clauses using proprietary vocabularies. Table 1 shows how a search for the term "obama" and an associated news page is specified for three selected sample repositories. The consequence of this heterogeneity is that hybrid queries written for a particular triple store are system-specific, making it hard to define such a query in a federated environment. Additionaly, a system has to deal with semantic heterogeneity such as, for instance, different scoring schemes for result ranking.

The second challenge concerns *efficient runtime processing* of hybrid queries in order to minimize the execution time. Optimal ordering of operators and the choice of processing techniques (e.g., nested loop join and symmetric hash join) depend on the selectivity of graph patterns and characteristics of the federated environment (e.g., hardware equipment and network latency of repositories). As a federation may include external data sources, collecting statistical information about remote sources may be infeasible (especially, if data is frequently updated). While there are heuristics for estimating the selectivity of SPARQL graph patterns using only static information (e.g., number of free variables, number of relevant data sources), estimating the selectivity of keyword search requests can be particularly difficult.

Finally, given that full-text and hybrid search queries often require only a subset of most relevant results, they represent a special case of top-k queries. Optimal processing techniques for such queries can be different from the ones retrieving complete result sets.

With this work we make the following novel contributions:

- We propose an extension to the SPARQL query algebra that allows to represent hybrid SPARQL queries in a triple-store-independent way (Section 3). On the basis of this algebra extension, we propose query optimization techniques to match keyword search clauses to appropriate repositories, combine retrieved results seamlessly, and reduce the processing time.

[1] http://www.w3.org/TR/sparql11-query/
[2] http://www.ontotext.com/owlim
[3] http://virtuoso.openlinksw.com/rdf-quad-store/
[4] http://dev.nepomuk.semanticdesktop.org/wiki/LuceneSail

- We propose novel runtime query execution techniques for optimized scheduling of tasks (Section 4), supporting on-the-fly adaptation of the query execution plan based on a cost model. These mechanisms allow for time-effective and robust execution of hybrid queries even in the absence of statistical data about federation members.
- We present and evaluate FedSearch (Section 5), which allows to process hybrid SPARQL queries efficiently in heterogeneous federations. Our evaluation based on two benchmarks shows substantial performance improvements achieved with static and runtime optimization mechanisms of FedSearch, sometimes reducing execution time by more than an order of magnitude.

2 Related Work

Processing queries in a federation of data sources has been studied for a long time in the database community [3,4]. Although this research forms the basis for approaches tackling distributed Linked Data sources, differences in data representation formats and access modes require special handling mechanisms. Existing systems divide into two categories depending on the assumed data access protocol: link traversal [5,6], where new sources are added incrementally by dereferencing URIs, and endpoint querying [1,2], which assume a set of known sources providing SPARQL endpoint services. While the former approach is targeted at open scenarios involving public Linked Data sources, the latter is more suitable for enterprise use cases that involve a set of internal repositories and combine their data with selected third-party ones.

The tasks of a query processing engine involve matching query clauses to relevant data sources, query optimization to find an optimal execution plan, and query execution aimed at minimizing the processing time. Default federation support in SPARQL 1.1[5] assumes explicit specification of graph patterns in a SERVICE clause, which are evaluated at the specified endpoint. Some systems go further and automatically determine relevant data sources for different query parts. For this purpose, SPLENDID [7] uses VoID[8] descriptors of federation members, while systems such as DARQ [9] utilize custom source profiles. Avalanche [10] does not require having data source statistics in advance, but gathers this information as part of the query optimization workflow. To avoid the need for statistical data about federation members, FedX [1] uses ASK queries to endpoints, while ANAPSID [2] utilises only schema-level information for source selection and uses sampling-based techniques to estimate selectivity and adaptive query processing to adjust the execution process on the fly. A substantial body of related work already exists on the topic of general SPARQL query optimization: e.g., in [11] an optimizer efficiently combining left-linear and bushy query plans is proposed. A good empirical comparison of the behavior of systems utilizing different join strategies is given by [12].

Keyword-based entity search over structured data represents a special case of semantic search and has been studied in parallel to structured query

[5] `http://www.w3.org/TR/sparql11-federated-query/`

processing (a survey of methods can be found in [13]). A natural evolution of purely keyword-based search involves hybrid search combining both paradigms. For processing such queries, Wang et al. [14] propose an extended ranking schema taking into account features from both full-text and structured data. Although existing approaches already provide complex and efficient query processing models, these techniques usually rely on detailed statistical information about both structured and unstructured data stored in the repositories. For this reason, we consider these methods complementary to our approach, which does not require such apriori information.

Finally, full-text and hybrid queries often require results to be ranked according to their relevance, while typically only the highest ranked ones are of interest for the user. For this reason, hybrid search queries represent a special case of top-k queries, in which the ranking function has to aggregate the ranking scores associated with keyword search results. The $\mathcal{S}$PARQL-$\mathcal{R}$ANK algebra [15] was proposed to enable static optimization of query plans containing ORDER and LIMIT modifiers. Our approach extends this algebra to incorporate full-text search clauses. A complementary approach proposed in [6] focuses on top-k query answering using link traversal for data access. This method features push-based processing of algebra operators instead of traditional pull-based techniques to reduce the effect of network latency issues and slow data sources.

3 Hybrid Search in SPARQL

Different triple stores use different syntax to express hybrid search SPARQL queries. In order to process such queries in a federation of heterogeneous data sources, a given query has to be tailored to the standards expected by each federation member. Given that keyword search clauses produce ordered result sets, the query engine must be able to adjust the query plan to retrieve top-k ranked query results in the most efficient way. To achieve this, our proposed approach involves abstracting from repository-specific syntax and expressing keyword search clauses in the query algebra in a uniform way. This section provides the necessary background information and discusses our extension of the SPARQL query algebra to represent hybrid queries and static query optimization techniques aimed at minimizing processing costs.

3.1 Basic Definitions

In a SPARQL query, the WHERE clause defines a *graph pattern* to be evaluated on an RDF graph G. An atomic graph pattern is a *triple pattern* defined as a tuple from $(I \cup L \cup V) \times (I \cup V) \times (I \cup L \cup V)$, where I, L, and V correspond to the sets of IRIs, literals, and variables respectively. Arbitrary graph patterns are constructed from triple patterns by means of JOIN, UNION, FILTER, and OPTIONAL operators. A *mapping* is defined as a partial function $\mu : V \rightarrow (I \cup L \cup B)$ (B is a set of blank nodes) [16], and the domain of the mapping $dom(\mu)$ expresses a subset of V on which the mapping is defined. Then, the semantics of SPARQL queries is expressed by means of a function $[\![P]\!]_G$, which

takes as input a graph pattern P and produces a set of mappings from the set of variables $var(P)$ mentioned in P to elements of the graph G. The binding of the variable $?x$ according to the mapping μ is denoted as $\mu(?x)$. The basic query algebra then defines the standard operations (Selection σ, Join $\bowtie$, Union $\cup$, Difference $\setminus$, and Left Join $⟕$) over the sets of mappings, and query evaluation involves translating the query into a *query tree* composed of these operations. For simplicity, in this paper we use the notation $P_1 \bowtie P_2$ to refer to the join operation over sets of mappings produced by the patterns P_1 and P_2.

In order to allow efficient processing of *top-k* queries, the $\mathcal{S}$PARQL-$\mathcal{R}$ANK algebra [15] introduces a new rank operator $\rho(P)$ which orders the set of input mappings according to some *scoring function* $\mathcal{F}$. The function $\mathcal{F}(b_1, \ldots, b_n)$ is defined over the set B of *ranking criteria* $b_i(?x_1, \ldots, ?x_m)$, where each ranking criterion specifies a function over the set of variables $var(P)$. Based on the semantics of the rank operator, the $\mathcal{S}$PARQL-$\mathcal{R}$ANK algebra proposes the rank-aware modifications of the standard combination operators (RankJoin $\bowtie^{\rho}$ and RankUnion U^{ρ}) and defines algebraic equivalences which can be used to reformulate and optimize the algebraic query tree, such as rank splitting, rank commutative law, and propagation of rank over union and join operations.

3.2 Background: FedX Federated SPARQL Query Engine

FedX [1] provides a framework for transparent access to data sources through a federation. It establishes a federation layer which employs several static and runtime optimization techniques. Static optimization includes reordering join operands with the aim of evaluating selective query parts first and executing filters early to reduce the size of intermediate results. At runtime FedX utilizes sophisticated join execution strategies based on distributed semijoins. One such strategy is the *Bind Nested Loop Join* (BNLJ) algorithm denoted by $\bowtie_{BNLJ}$ – a variation of the block nested loop join, in which each subquery sent to a remote data source probes it at once for several partial mappings pulled from the left operand. This significantly reduces the number of required remote requests. In addition, FedX applies pipelining to compute results as fast as possible: a special scheduler maintains a queue of atomic operations, and processes them in parallel. Instead of waiting for execution of each subquery in sequence, the system sends them in parallel and collects results as soon as they arrive, which further improves the execution performance.

The system further identifies situations where a query can be partitioned into so-called exclusive groups Σ_{excl}, which combine several triple patterns that can be evaluated together on the same data source. All these optimization techniques are applied automatically and do not require any interaction with the user. An important feature of FedX is its independence from statistical data about the federation members. Instead of relying on indexes or catalogs to decide on the relevance of a source, FedX uses caching in combination with SPARQL ASK queries. In this way it allows for on-demand federation setup (meaning that data sources can be added and removed from the federation at query time). Our extension of FedX – FedSearch – maintains this property.

3.3 Hybrid Search in SPARQL Algebra

To enable hybrid queries without modifying the SPARQL syntax, existing triple stores express keyword search using special graph patterns which use proprietary vocabularies. At evaluation time, the query engine recognizes these special terms, extracts the search parameters (keywords and projected variables), evaluates the keyword search using its full-text index, and returns a set of mappings binding the projected variables to search answers and their properties (related resource, matched value, relevance score). Thus, graph patterns defining search parameters do not follow the SPARQL semantics, as their result sets are in general not equivalent to the result of algebra operations combining the mapping sets of their constituting triple patterns. This has strong implications for federated query processing, as triple patterns related to keyword search cannot be evaluated separately either on the same or different federation members. Such proprietary graph patterns have to be recognized by the query engine, isolated, and evaluated as whole blocks.

For this purpose, FedSearch introduces the notion of a *keyword search group* as a special graph pattern in the query tree.

***Definition 1**: A keyword search group Σ^{KS} is a tuple (q, v, r, s, p, sn) defined as follows:*

- $q \in L$ – *a literal value representing the keyword query*
- $v \in (V \cup \{nil\})$ – *a variable bound to a literal value matching the keyword*
- $s \in (I \cup V)$ – *a subject resource connected to v.*
- $p \in (I \cup V \cup \{nil\})$ – *a property connecting s to v*
- $r \in (V \cup \{nil\})$ – *a variable bound to a literal value between 0 and 1 representing a normalized keyword search score (1 corresponding to the highest degree of relevance)*
- $sn \in (V \cup \{nil\})$ – *a value snippet highlighting the matching keywords*

The value *nil* provided for a tuple element implies that the corresponding value or variable does not need to be included in the query: e.g., the queries shown in Table 1 do not explicitly project the relevance score.

Some of these elements are source-dependent: e.g., not all data repositories can provide the value snippet (a standard feature of LuceneSail, but not available in OWLIM), or, more importantly, returned score values cannot be compared across different data sources, even those of the same type. Traditionally, methods for combining ranked search results [17,18] primarily rely on the analysis of matched values and re-estimation of their relevance to the query string. This procedure, however, is too costly in the context of SPARQL query processing, as it requires additional downloading, parsing, and processing of whole matched values. Thus, meaningful ranking of the combined result set according to some common relevance criterion is impossible without knowing the statistics of back-end repositories.

For this reason, FedSearch operates over normalized query scores lying in the interval $[0, \ldots, 1]$. It also applies the algebra operators *RankUnion* and *RankJoin*.

The RankUnion operation over normalized scores (1) preserves the order of results retrieved from the same source and (2) ensures that results from one source do not suppress results from another source due to different scales. To combine ranking scores from different keyword search groups, the RankJoin operation applies the function $\mathcal{F}(r_1, r_2) = avg(r_1, r_2)$. This function preserves monotonicity of the result ranking with respect to the original scores (i.e., if $(r_1[\mu_1] < r_1[\mu_2])$ AND $(r_2[\mu_1] < r_2[\mu_2]) \Rightarrow (\mathcal{F}(r_1[\mu_1], r_2[\mu_1]) < \mathcal{F}(r_1[\mu_2], r_2[\mu_2]))$, while also taking both scores into account and maintaining the original scale.

3.4 Static Query Optimization

FedSearch assumes that the user's query is expressed using the vocabulary supported by one of the federation members. By default, the parsed query tree only consists of basic SPARQL operations applied to atomic triple patterns: for example, Figure 1 shows the initial plan for the example query from Table 1 expressed in LuceneSail syntax. The original FedX system applies static query optimization techniques aimed at adjusting the given query to the federated environment: matching triple patterns to relevant sources, combining together the exclusive groups of triple patterns, reordering join operands according to their estimated selectivity.

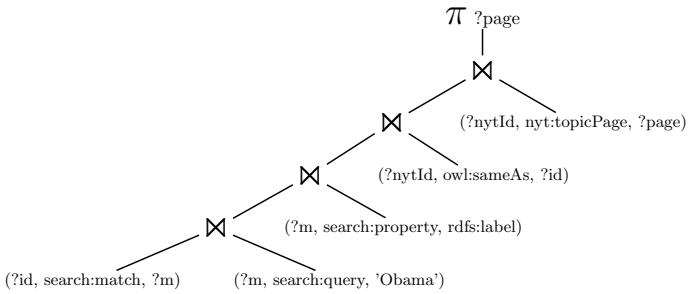


Fig. 1. Unoptimized hybrid search query tree

To process a hybrid query, the task of the static optimization stage includes three additional subtasks:

Detecting and Isolating Keyword Search Groups. At this stage, the query optimizer selects and groups triple patterns, which together form keyword search groups. In the query tree, these triple patterns are replaced with a single Σ^{KS} pattern. The result of this stage is an abstract query tree independent of concrete triple stores.

Mapping Keyword Search Groups to Relevant Data Sources. Unless the target is given in the SERVICE clause, each Σ^{KS} can potentially produce mappings from any data source supporting full-text search. Accordingly, the Σ^{KS} is replaced with the grounded repository-dependent graph

pattern Σ_g^{KS}, which is associated with all endpoints of the same type (LuceneSail, Virtuoso, etc) and contains corresponding source-dependent triple patterns. The federation configuration contains the backend repository type of its members. If the federation includes repositories of several types, the keyword search group is replaced with a union of several grounded keyword search groups. The result of this stage is a grounded query tree.

Modifying the Query Tree to Take Result Ranking into Account. Each keyword search graph pattern Σ_g^{KS} is expanded to return the score value r_i, if it does not project the relevance score explicitly,

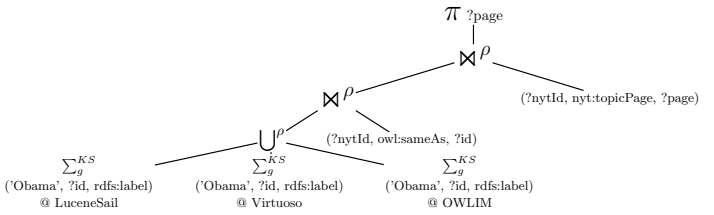


Fig. 2. Grounded and optimized hybrid search query tree

The resulting expanded query tree is further processed to enforce the ordering of final results according to the combined score $\mathcal{F}(\{r_i\})$ and to minimize the query execution time. For this purpose, equivalence relations defined in the $\mathcal{S}$PARQL-$\mathcal{R}$ANK algebra are applied:

- Partial ranking criteria r_i of keyword search clauses are propagated towards the root of the query tree. This involves converting the standard Union and Join operations to corresponding *RankUnion* and *RankJoin* according to the rules defined in [15]. Relevance scores are combined using normalization and averaging, as discussed in section 3.3.
- Top-level ordering criteria (if defined) are propagated down the query tree so that atomic clauses produce their mapping sets already ordered.
- LIMIT thresholds are moved towards the leaves of the tree using the relation $SLICE(P_1 \cup P_2, lim) = SLICE(SLICE(P_1, lim) \cup SLICE(P_2, lim), lim)$. This reduces the costs of local evaluation of keyword search clauses as well as network resources for transferring result sets.

Figure 2 shows the result of the static optimization operations applied to the example query from Table 1 for a federation including repositories of three types: OWLIM, Virtuoso, and LuceneSail.

4 Optimizing Top-k Hybrid Query Execution

Although static query optimization already helps to reduce the expected execution time, the actual performance strongly depends on the way the operators

(primarily, joins) are processed. The *Bind Nested Loop Join* technique of the original FedX system significantly reduces the number of required remote requests by grouping together several binding sets in one probing request and using pipelining.

For processing top-k queries and hybrid queries in particular, however, this mechanism is insufficient for several reasons:

- Optimal scheduling of remote requests can differ for top-k queries and queries without a LIMIT modifier. For top-k queries it is important to produce the first complete results as soon as possible, even at the cost of some extra synchronization time, as it can possibly make processing of low-ranked partial results unnecessary.
- More importantly, performance strongly depends on the order of operands. In case if one operand is more selective than the other, reversing their order leads to big differences in execution time. While static optimization tries to sort the join operands according to the expected selectivity, there is no way to estimate selectivity of keywords in the general case.

To deal with these issues, we apply runtime join processing optimization techniques: *synchronization of loop join requests* and adaptive *parallel competing join processing* for queries containing several ordered clauses.

4.1 Synchronization of Loop Join Requests

As an example, let us consider a hybrid search query, which searches for all drugs interacting with aspirin and their side effects, while taking input in different languages:

```
  SELECT ?drugName ?sideEffect WHERE {
1   ?val luc:luceneIndex "acetylsalicylsäure" .        //DBpedia
2   ?id1 rdfs:label ?val .
3   ?id2 owl:sameAs ?id1 .                             //DrugBank
4   ?interaction drugbank:interactionDrug1 ?id2 .
5   ?interaction drugbank:interactionDrug2 ?id3 .
6   ?id3 rdfs:label ?drugName .
7   ?id3 owl:sameAs ?id4 .
8   ?id4 sider:sideEffect ?sideEffectId .              //SIDER
9   ?sideEffectId rdfs:label ?sideEffect .
  }
```

This query involves combining data from 3 sources: DBpedia[6] (triple patterns 1-2), DrugBank[7](3-7), and SIDER[8](8-9). During the static optimization stage these triple patterns are combined into 3 groups, which we denote as Σ_1^{KS} (DBpedia), Σ_2 (DrugBank), and Σ_3 (SIDER). When performing a bind nested loop join, the algorithm will iterate through the mapped tuples μ_i of the Σ_1^{KS} result

[6] http://dbpedia.org
[7] http://wifo5-04.informatik.uni-mannheim.de/drugbank/
[8] http://wifo5-04.informatik.uni-mannheim.de/sider/

set and probe the second operand Σ_2 binding the variable $?id1$. While gradually receiving results μ_{ij} from $(\Sigma_1^{KS} \bowtie_{BNLJ} \Sigma_2)$ and iterating through them, the last operand Σ_3 will be joined using the mappings $\mu_{ij}(?id4)$. As described in [1], this process is parallelized so that each probing subquery in the nested loop is scheduled in a processing queue and then sent in a separate thread. However, depending on the scheduling approach, the process can be performed in two ways:

- *Breadth-first*: In this way, all probing subqueries will be immediately added to the processing queue. Thus, the executor will first send all subqueries for $\Sigma_2(\mu_i(?id1))$ and only then, while results are arriving, send the subqueries for $\Sigma_3(\mu_{ij}(?id4))$.
- *Depth-first*: In this way, when the results from $\Sigma_2(\mu_i(?id1))$ begin to arrive, and subqueries for $\Sigma_3(\mu_{ij}(?id4))$ are added to the queue, the executor immediately moves them to the start of the queue, even if not all $\Sigma_2(\mu_i(?id1))$ requests have been sent yet.

Depending on the type of the query, FedSearch decides on using either of the two techniques. For top-k queries, the depth-first technique is applied. This involves additional synchronization costs to manipulate the task queue and maintain the result set ordering: results returned by probing subqueries to the operands Σ_2 and Σ_3 must be processed in the same order as they were sent. However, the depth-first approach allows receiving first complete results early and potentially terminate the processing early after k results are collected. On the contrary, the breadth-first approach gives an advantage when a complete result set is required: because all nested loops have to be executed completely, extra synchronization handling is unnecessary.

4.2 Adaptive Processing of Rank-Join Operators: Parallel Competing Joining

If a query contains more than one keyword search clause, it is impossible to determine the more selective one without possessing the distribution statistics of keywords. As a result, the join sequence determined at the static optimization stage can lead to a non-optimal execution plan. To avoid this, in the following we present *parallel competing rank join processing*, a novel technique which allows on-the-fly adaptation of the query plan at execution time.

Processing *N*-ary Join. The high-level idea of this technique is to use a subset of the join operands as *seeds* to allow adaptive query processing. In particular, competing join plans for those operands that determine the ordering are executed in parallel – thus competing against each other – while the other join operands are computed iteratively using the intermediate results from the seeds as soon as they arrive. Whenever an iteration completes with processing its partial result set, a re-evaulation of all query plans takes place to ensure that the next operand is joined to the most selective seed. Finally, the ordered intermediate result

sets of competing join plans are combined using the N-ary Pull/Bound Rank Join algorithm (PBRJ) [19], which produces the results ranked according to the aggregated scores of its operands.

Algorithm 1 a) depicts our *Parallel Competing Rank Join* technique. Given the set of ranked join operands P^ρ (including all Σ^{KS} groups) – the *seeds* – and the set of unranked operands P^u, our algorithm first determines suitable competing join plans and then executes each competing *seed* P_i^ρ in parallel. The incoming intermediate results are processed and joined using the cost-based adaptive query technique explained below, yielding ordered results sets for each competing join plan. To combine the partial result sets produced by all seeds, FedSearch uses the N-ary PBRJ variant which modifies the symmetric hash join technique to process RankJoin in an efficient way.

Algorithm 1. Adaptive Processing of Rank Joins

a) Parallel Competing Rank Joins

```
1: P^ρ: ranked operands (incl. all Σ^KS)
2: P^u: unranked operands
3: P_left ← P^u
4: for all P_i^ρ ∈ P^ρ do
5:     Q_i ← joinOrderSort(P^u)
6:     P_i ← P_i^ρ
7:     start(P_i)
8: ...
9: if P_left = ∅ then
10:    return PBRJ({P_i})
```

b) Processing Incoming Results

```
1: procedure pushResults(P_curr, ⟦P_curr⟧_G)
2:     P^u_next ← Q_curr.next
3:     c = cost(P_curr ⋈_BNLJ P^u_next)
4:     for all P_i ≠ P_curr do
5:         pos = Q_i.indexOf(P^u_next)
6:         c_i = costLeft(P_i)
7:             + Σ_{j=1}^{pos} cost(P_i ⋈_BNLJ P^u_j)
8:             + cost(P_i ⋈_BNLJ P^u_next)
9:         if c_i < c then
10:            return
11:    P_curr ← P_curr ⋈_BNLJ P^u_next
12:    for all Q_i do
13:        Q_i ← Q_i \ {P^u_next}
14:    P_left ← P^left \ {P^u_next}
15:    start(P_curr)
```

The processing step of incoming intermediate results is depicted in Algorithm 1 b). Whenever a seed-computation has received the complete result set $[\![P_{curr}]\!]_G$ for its current operation, a re-evaluation of the execution plans takes place. The re-evaluation procedure estimates the cost c of executing the join $P_{curr} \bowtie P^u_{next}$ and compares it to the respective costs c_i of joining the operand P^u_{next} as part of other *"competitor"* query plans. The cost model used in Algorithm 1 b) is described in detail in the following.

Estimating Join Cost. The basis for our cost model is the average request time of a *Bind Nested Loop Join* (BNLJ) operation at a remote SPARQL service. In general, these execution times can differ substantially for two different queries over the same endpoint. However, FedSearch only estimates the cost of BNLJ subqueries over single triple patterns, which have similar access times. For that reason, FedSearch keeps the statistics of executing BNLJ requests including the average execution time for the data source $\tau_{avg}(s)$ and the average execution time over all sources τ_{avg}. Note that the latter value is used instead of $\tau_{avg}(s)$ if for some source s there is no sufficient historical data.

Cost of a future join. In Algorithm 1, the cost of a join is estimated based on the average request time and the known cardinality of the received result set according to the following formula:

$$cost(P_{rec} \bowtie_{BNLJ} P^u_{next}) = \frac{(N(|[\![P_i]\!]_G|) - 1) * \tau_{avg}(s)}{N_{thread}/|\mathcal{P}^\rho|} + \tau_{avg}(s)$$

Here $N(|[\![P_i]\!]_G|)$ denotes the number of probing subqueries where according to BNLJ each subquery holds bindings for multiple mappings $\mu \in [\![P_i]\!]_G$. If the operand P^u_{next} has multiple data sources, we use the maximal average execution time of all sources, denoted by $max_s(\tau_{avg}(s))$. Finally, N_{thread} holds the number of parallel worker threads used by the system, which means that processing one of the competing query plans can utilize on average $N_{thread}/|\mathcal{P}^\rho|$ threads, where $\mathcal{P}^\rho$ denotes the set of ranked operands (i.e., the seeds).

For non-atomic operands involving other joins and unions, the cost is determined as follows:

- if the operand is a union, the cost is determined by the maximum cost of the individual union operands multiplied by a coefficient w that estimates the additional cost of combining the results.
- if the operand is a join, left-join or difference, the cost is determined by the sum of the individual costs of the join operands.

Estimating costs of competing branches. The cost c_i of joining the next operand P^u_{next} as part of a join sequence Q_i, which competes with the current sequence Q_{curr}, consists of two components:

1. the remaining cost of the current operation, denoted by $costLeft(P_i)$.
2. the cost of joining P_i with all operands in Q_i until P^u_{next} (inclusive): $\sum_{j=1}^{pos} cost(P_i \bowtie_{BNLJ} P^u_j) + cost(P_i \bowtie_{BNLJ} P^u_{next})$

The remaining cost is only considered if the operation is running, and can be estimated as $costLeft(P_k) = \tau_{passed} * (\frac{N_{total}}{N_{received}} - 1)$, where τ_{passed} is the time since the start of the operation, N_{total} is a total number of subqueries sent, and $N_{received}$ is the number of subqueries for which results have already been received.

Decision on joining the next operand. Depending on the computed costs for the competing execution sequences Q_i, FedSearch decides to either execute a sequence or reject it. If the cost c is found to be lower than all competitors' costs, the respective execution sequence continues. Otherwise, the execution of the current sequence Q_{curr} is considered rejected in favour of a competing plan Q_{comp}. Note that a rejected execution plan can be re-initiated, if during processing of Q_{comp} its cost is re-estimated to be higher than Q_{curr}, and the operand P^u_{next} has not been joined as a part of Q_{comp} yet.

In this way, multiple ranked operands are processed in an efficient way: due to the cost estimation process and the fact that more selective seed queries usually return results earlier, new operands are naturally joined to the more selective seed, thus minimizing the number of required nested loop join subqueries.

5 Evaluation

To validate the FedSearch approach, we performed experiments with two different benchmark datasets. First, we reused the LUBMft query benchmark proposed in [20] for evaluating full-text search performance of RDF triple stores. The benchmark extends the well-known LUBM university benchmark dataset [21] with full-text data and includes a comprehensive set of full-text and hybrid SPARQL queries testing various performance aspects. Second, we reused the set of Life Sciences data sources from the FedBench benchmark for federated query processing [22]. Since FedBench does not include hybrid search queries, we have extended the query set with 6 additional queries involving full-text search clauses. In both sets of experiments target endpoints were hosted as separate OWLIM repositories on a Windows server with two 3GHz Intel processors and 20GB of RAM. We compared the runtime query processing techniques of FedSearch with two other systems: the original FedX architecture and ARQ-$\mathcal{R}$ANK, the open source implementation of SPARQL-$\mathcal{R}$ANK algebra provided by its authors[9]. The original FedX architecture made use of the static optimization techniques described in section 3.4 (so that full-text search clauses could be matched to appropriate sources), but not the runtime optimization. For ARQ-$\mathcal{R}$ANK, which cannot automatically determine relevant data sources, the queries were expanded so that each graph pattern was explicitly targeted at relevant endpoints using SERVICE clauses. Each query was executed 10 times, out of which the first 5 queries were considered "warm-up" to fill the relevant endpoint caches, while the result was equivalent to the average over remaining 5 runs. Benchmark queries, the complete results of the tests, as well as a downloadable version of FedSearch are available online on our web site[10].

5.1 LUBMft Benchmark

To perform tests with the LUBMft benchmark dataset, it has been split into 6 parts which represented different endpoints: generated dump files were distributed equally between endpoints resulting in a horizontal partitioning. The benchmark includes 24 queries aimed at testing different triple store capabilities related to keyword search. We used only the first 14 queries covering pure full-text search and hybrid search. Out of these, 8 queries contain only keyword search clauses, while 6 queries are hybrid: 3 queries containing 1 keyword search clause, 2 queries with 2 clauses, and 1 query with 3 keyword search groups. Table 2 shows the average query processing times achieved on the largest LUBMft dataset ($N = 50$)[11]. For all values of k FedSearch achieved the best overall performance. For pure full-text queries (*q1.1 - q4*) performance of all three systems is similar when the complete result set is required. However, for

[9] `http://sparqlrank.search-computing.org/`

[10] `http://fedsearch.fluidops.net/resource/FedSearch`

[11] Queries 2.1, 2.2, 5.1, and 5.2 are skipped due to the lack of space, as they are largely redundant with respect to 1.1 and 1.2. However, they are included in our complete result set available online as well as results for $N = 1, 5, 10$.

Table 2. Average execution time (sec) for LUBMft queries (N=50) taken over 5 query runs. Numbers in brackets indicate the number of runs which resulted in timeout.

k	System	q1.1	q1.2	q3	q4	q6	q7	q8	q9	q10	q11	Geom. Mean
$N_{answers}$		1933	51257	52784	51	1933	1933	704	60	2783	15	
all	FedSearch	0.33	12.51	13.39	0.04	4.96	9.93	13.42	13.36	22.58	14.20	4.75
	FedX	0.29	12.54	13.22	0.02	5.53	6.58	18.87	110.48	455.19 (2)	396.79 (2)	10.24
	ARQ-$\mathcal{R}$ANK	0.62	12.72	13.01	0.22	32.79	66.65	169.34	142.75	Timeout	Timeout	13.63
100	FedSearch	0.08	0.11	0.10	0.04	0.80	1.23	5.01	10.22	26.71	15.48	0.96
	FedX	0.35	11.53	13.34	0.03	4.22	8.79	17.82	123.07	459.90 (3)	106.92 (3)	9.41
	ARQ-$\mathcal{R}$ANK	0.58	1.60	5.07	0.22	2.16	4.79	18.84	142.75	Timeout	Timeout	3.60
10	FedSearch	0.03	0.34	0.04	0.03	0.58	0.73	3.55	11.88	24.25	16.71	0.78
	FedX	0.29	11.56	12.51	0.27	4.31	7.76	18.26	144.22	456.18	153.83 (3)	12.15
	ARQ-$\mathcal{R}$ANK	0.57	1.59	1.52	0.22	0.67	0.98	1.66	142.75	Timeout	Timeout	1.62
1	FedSearch	0.02	0.04	0.04	0.02	0.74	0.72	1.82	11.47	23.95	15.09	0.54
	FedX	0.43	11.54	12.93	0.02	4.21	8.03	17.94	135.80	455.32 (3)	398.25 (3)	10.42
	ARQ-$\mathcal{R}$ANK	0.57	1.59	1.52	0.22	0.54	0.63	0.41	142.75	Timeout	Timeout	1.25

top-k queries applying static optimization (pushing the limit modifier to atomic clauses) reduces the cost of remote evaluation and result set transfer over the network. FedSearch further improves on this due to parallelization. Hybrid queries with a single FTS clause (*q6*-*q8*) demonstrate respective benefits of depth-first and breadth-first N-ary join processing: the former gives an advantage when executing *top-k* queries (FedSearch and ARQ-$\mathcal{R}$ANK outperform FedX) while the latter is preferred when a complete result set is required. Finally, queries *q9*-*q11*, which contain two or more FTS clauses, illustrate the benefits of the parallel competing rank join algorithm. Results obtained for FedSearch and two other systems differ sometimes by more than one order of magnitude, while FedSearch delivers more robust performance: evaluation time does not depend on the join order produced at the static optimization stage.

5.2 FedBench Life Sciences Benchmark

The Life Sciences module of the FedBench benchmark includes 4 datasets containing medicine-related data: KEGG[12], DrugBank[13], ChEBI[14], and a subset of DBpedia[15]. To test hybrid query performance we used a set of 6 queries, which we constructed with the following requirements:

- Each query requires accessing at least 3 datasets from the federation.
- Queries include different proportion of full-text vs graph clauses: 2 queries are full-text only, 2 queries are hybrid with 1 full-text search clause, and 2 queries are hybrid with 2 full-text search clauses.
- Full-text search clauses have different degrees of selectivity.

Evaluation results with these queries are shown in Table 3 (due to smaller size of result sets, we only performed experiments with $k = 10$). The scale of

[12] `http://www.genome.jp/kegg/kegg1.html`
[13] `http://wifo5-04.informatik.uni-mannheim.de/drugbank/`
[14] `http://www.ebi.ac.uk/chebi/userManualForward.do`
[15] `http://dbpedia.org`

differences between processing engines is smaller than in case of horizontal partioning: mainly because triple patterns with bound predicates do not need to be evaluated on all endpoints, which reduces the overall number of required remote requests. However, the results are largely consistent with the LUBMft experiments. For pure full-text *top-k* search, applying static *top-k* optimization leads to substantial performance improvement if the overall result set is large. For hybrid queries with a single keyword search clause using depth-first N-ary join processing reduces execution time (ARQ-$\mathcal{R}$ANK even marginally outperforms FedSearch due to "fixed costs" of static optimization), while, however, it becomes a drawback when a complete result set is required. Finally, for hybrid queries with multiple FTS clauses the parallel competing bound join algorithm provides a clear advantage.

Table 3. Average execution time (sec) for Life Science queries taken over 5 query runs

k	System	q1	q2	q3	q4	q5	q6	Geom. Mean
$N_{answers}$		8129	57	255	930	15	22	
all	FedSearch	0.50	0.09	0.55	6.20	2.33	7.40	1.17
	FedX	0.72	0.03	0.66	6.42	8.47	28.53	1.62
	SPARQL-RANK	0.95	0.24	3.24	32.10	4.56	21.54	3.64
10	FedSearch	0.06	0.03	0.74	0.81	2.36	7.85	0.50
	FedX	0.78	0.02	0.70	6.34	5.30	38.32	1.57
	SPARQL-RANK	0.07	0.01	0.12	0.42	3.73	21.41	0.39

6 Conclusion and Outlook

In this paper, we proposed novel static and runtime optimization techniques as a means to enable processing hybrid search queries in a federation of SPARQL endpoints. The evaluation of our implemented system, FedSearch, has shown that it allows for substantial reduction of processing time without relying on statistical data about the content of federation members.

One immediate practical benefit provided by FedSearch is the possibility to realize data access to a diverse set of sources including different triple stores and full-text indices through a common access interface. As a future direction of work, we are planning to utilize this ability to support practical use cases requiring end-user applications to consume data stored in multiple data sources in a seamless way.

While the ability to establish on-demand federation without significant additional effort is a requirement for our system, existing statistical data (e.g., VoID descriptors) can be utilized to further improve its performance. Employing additional techniques to estimate keyword selectivity (e.g., based on [23]) constitutes another promising direction.

References

1. Schwarte, A., Haase, P., Hose, K., Schenkel, R., Schmidt, M.: FedX: Optimization techniques for federated query processing on linked data. In: Aroyo, L., Welty, C., Alani, H., Taylor, J., Bernstein, A., Kagal, L., Noy, N., Blomqvist, E. (eds.) ISWC 2011, Part I. LNCS, vol. 7031, pp. 601–616. Springer, Heidelberg (2011)

2. Acosta, M., Vidal, M.-E., Lampo, T., Castillo, J., Ruckhaus, E.: ANAPSID: An adaptive query processing engine for SPARQL endpoints. In: Aroyo, L., Welty, C., Alani, H., Taylor, J., Bernstein, A., Kagal, L., Noy, N., Blomqvist, E. (eds.) ISWC 2011, Part I. LNCS, vol. 7031, pp. 18–34. Springer, Heidelberg (2011)
3. Sheth, A.P.: Federated database systems for managing distributed, heterogeneous, and autonomous databases. In: VLDB 1991, p. 489 (1991)
4. Kossmann, D.: The state of the art in distributed query processing. ACM Computing Surveys 32(4), 422–469 (2000)
5. Hartig, O.: Zero-knowledge query planning for an iterator implementation of link traversal based query execution. In: Antoniou, G., Grobelnik, M., Simperl, E., Parsia, B., Plexousakis, D., De Leenheer, P., Pan, J. (eds.) ESWC 2011, Part I. LNCS, vol. 6643, pp. 154–169. Springer, Heidelberg (2011)
6. Wagner, A., Duc, T.T., Ladwig, G., Harth, A., Studer, R.: Top-k linked data query processing. In: Simperl, E., Cimiano, P., Polleres, A., Corcho, O., Presutti, V. (eds.) ESWC 2012. LNCS, vol. 7295, pp. 56–71. Springer, Heidelberg (2012)
7. Görlitz, O., Staab, S.: Splendid: Sparql endpoint federation exploiting void descriptions. In: COLD 2011, at ISWC 2011 (2011)
8. Alexander, K., Cyganiak, R., Hausenblas, M., Zhao, J.: Describing linked datasets - on the design and usage of void. In: LDOW 2009 (2009)
9. Quilitz, B., Leser, U.: Querying Distributed RDF Data Sources with SPARQL. In: Bechhofer, S., Hauswirth, M., Hoffmann, J., Koubarakis, M. (eds.) ESWC 2008. LNCS, vol. 5021, pp. 524–538. Springer, Heidelberg (2008)
10. Basca, C., Bernstein, A.: Avalanche: Putting the spirit of the web back into Semantic Web querying. In: SSWS 2010 Workshop (2010)
11. Vidal, M.-E., Ruckhaus, E., Lampo, T., Martínez, A., Sierra, J., Polleres, A.: Efficiently joining group patterns in SPARQL queries. In: Aroyo, L., Antoniou, G., Hyvönen, E., ten Teije, A., Stuckenschmidt, H., Cabral, L., Tudorache, T. (eds.) ESWC 2010, Part I. LNCS, vol. 6088, pp. 228–242. Springer, Heidelberg (2010)
12. Montoya, G., Vidal, M.-E., Corcho, O., Ruckhaus, E., Buil-Aranda, C.: Benchmarking federated sparql query engines: Are existing testbeds enough? In: Cudré-Mauroux, P., Heflin, J., Sirin, E., Tudorache, T., Euzenat, J., Hauswirth, M., Parreira, J.X., Hendler, J., Schreiber, G., Bernstein, A., Blomqvist, E. (eds.) ISWC 2012, Part II. LNCS, vol. 7650, pp. 313–324. Springer, Heidelberg (2012)
13. Tran, T., Mika, P.: Semantic search - systems, concepts, methods and the communities behind it. Technical report
14. Wang, H., Tran, T., Liu, C., Fu, L.: Lightweight integration of IR and DB for scalable hybrid search with integrated ranking support. Journal of Web Semantics 9(4), 490–503 (2011)
15. Magliacane, S., Bozzon, A., Della Valle, E.: Efficient execution of top-K SPARQL queries. In: Cudré-Mauroux, P., Heflin, J., Sirin, E., Tudorache, T., Euzenat, J., Hauswirth, M., Parreira, J.X., Hendler, J., Schreiber, G., Bernstein, A., Blomqvist, E. (eds.) ISWC 2012, Part I. LNCS, vol. 7649, pp. 344–360. Springer, Heidelberg (2012)
16. Pérez, J., Arenas, M., Gutierrez, C.: Semantics and complexity of SPARQL. ACM TODS 34(3) (2009)
17. Craswell, N., Hawking, D., Thistlewaite, P.B.: Merging results from isolated search engines. In: Australasian Database Conference (1999)
18. Si, L., Callan, J.: A semisupervised learning method to merge search engine results. ACM Transactions on Information Systems 21(4), 457–491 (2003)
19. Schnaitter, K., Polyzotis, N.: Optimal algorithms for evaluating rank joins in database systems. ACM Transactions on Database Systems 35(1) (2008)

20. Minack, E., Siberski, W., Nejdl, W.: Benchmarking fulltext search performance of RDF stores. In: Aroyo, L., Traverso, P., Ciravegna, F., Cimiano, P., Heath, T., Hyvönen, E., Mizoguchi, R., Oren, E., Sabou, M., Simperl, E. (eds.) ESWC 2009. LNCS, vol. 5554, pp. 81–95. Springer, Heidelberg (2009)
21. Guo, Y., Pan, Z., Heflin, J.: LUBM: A benchmark for OWL knowledge base systems. Journal of Web Semantics 3, 158–182 (2005)
22. Schmidt, M., Görlitz, O., Haase, P., Ladwig, G., Schwarte, A., Tran, T.: FedBench: A benchmark suite for federated semantic data query processing. In: Aroyo, L., Welty, C., Alani, H., Taylor, J., Bernstein, A., Kagal, L., Noy, N., Blomqvist, E. (eds.) ISWC 2011, Part I. LNCS, vol. 7031, pp. 585–600. Springer, Heidelberg (2011)
23. Wagner, A., Bicer, V., Tran, T.D.: Selectivity estimation for hybrid queries over text-rich data graphs. In: EDBT 2013, pp. 383–394. ACM, New York (2013)

Getting Lucky in Ontology Search: A Data-Driven Evaluation Framework for Ontology Ranking

Natalya F. Noy, Paul R. Alexander, Rave Harpaz,
Patricia L. Whetzel, Raymond W. Fergerson, and Mark A. Musen

Stanford Center for Biomedical Informatics Research
Stanford University, Stanford, CA, 94305, USA
{noy,palexander,rharpaz,whetzel,rayferg,musen}@stanford.edu

Abstract. With hundreds, if not thousands, of ontologies available today in many different domains, ontology search and ranking has become an important and timely problem. When a user searches a collection of ontologies for her terms of interest, there are often dozens of ontologies that contain these terms. How does she know which ontology is the most relevant to her search? Our research group hosts BioPortal, a public repository of more than 330 ontologies in the biomedical domain. When a term that a user searches for is available in multiple ontologies, how do we rank the results and how do we measure how well our ranking works? In this paper, we develop an evaluation framework that enables developers to compare and analyze the performance of different ontology-ranking methods. Our framework is based on processing search logs and determining how often users select the top link that the search engine offers. We evaluate our framework by analyzing the data on BioPortal searches. We explore several different ranking algorithms and measure the effectiveness of each ranking by measuring how often users click on the highest ranked ontology. We collected log data from more than 4,800 BioPortal searches. Our results show that regardless of the ranking, in more than half the searches, users select the first link. Thus, it is even more critical to ensure that the ranking is appropriate if we want to have satisfied users. Our further analysis demonstrates that ranking ontologies based on page view data significantly improves the user experience, with an approximately 26% increase in the number of users who select the highest ranked ontology for the search.

1 "I'm Feeling Lucky" in Ontology Search

Consider a user who needs to find an ontology to use as a source of terms to annotate descriptions of clinical trials. She searches a library of ontologies [1], such as BioPortal, a public repository of more than 300 biomedical ontologies and terminologies [2]. She puts in a term "myocardial infarction"—her subject of interest. She receives 149 results in 32 ontologies. Twenty two ontologies contain a class named precisely "myocardial infarction" (with variation only in capitalization); other results have this phrase as synonyms of the class name, or have

H. Alani et al. (Eds.): ISWC 2013, Part I, LNCS 8218, pp. 444–459, 2013.

it in a property value. If our user is not familiar with the ontologies, how does she know which one of the 22 ontologies to use? Which one does everybody else use? Which one has more information about the terms that she is interested in? Naturally, to answer this question perfectly, we must know much more than our user's search term. It would help to know which task she is trying to achieve (e.g., annotation of text), what are her preferred ontologies, whether or not she requires conformance to specific standards, and so on. However, in many cases, we do not have this information; when a user searches an ontology library, the only information that we often have is the user's search term—and we must produce the best ranking of results based only on this information.

Ontology researchers have addressed the problem of ontology selection and ranking over the years. They have proposed a number of algorithms, which take into account the ontologies themselves, the search terms, and the repository as a whole. We review some of these approaches in Section 2. Researchers evaluated these approaches in small-scale user studies with hand-selected users.

In this paper, we propose a framework for evaluating the effectiveness of ontology ranking by using search logs. We analyze the position of the ontologies that the user selects after an ontology-search engine presents her with the search results. We use the position of that selection among the search results as a measure of the effectiveness of a ranking algorithm: the closer the user's selection is to the top-ranked result, the better the algorithm worked for this user. Our goal is to achieve a ranking in which most users feel "lucky" by following the top link, just as many of us do with Web search engines (e.g., Google and Bing). We evaluate our approach by using extensive search logs from the users who perform search on the BioPortal site over a period of several months. Specifically, this paper makes the following contributions:

- We propose a data-driven framework for evaluating ontology ranking based on user search logs.
- We propose several features for ontology ranking based on user behavior in BioPortal, an open community-based ontology repository. These features include pageviews, web service calls, comments left on the site, and others.
- We use our data-driven framework to evaluate the effect of different features on the ontology ranking based on search logs from four months of BioPortal searches (4,859 by users from 969 unique IP addresses).

2 Related Work in Ontology Ranking and Evaluation

The problem of finding the "best" ontology in response to a user's search consists of two main components: (1) *selecting* relevant ontologies from a collection and (2) *ranking* the results to present the most relevant ontologies first.

Over the past decade, researchers have developed many algorithms for selecting ontologies that are relevant to a user query. These algorithms use description logic reasoning [3], corpus analysis [4,5], graph matching [6] and other approaches in order to find the relevant ontologies. When traditional retrieval

methods do not return sufficient results, algorithms use *query expansion* based on the hierarchy in the ontology [7], lexical-semantic relations [8], or statistical analyses [9]. In many cases, terms in more than one ontology match the user query, and therefore, we must rank the results in a way that we believe to be most meaningful to the user [10]. Researchers have explored links between ontologies [11], structure-based ranking [12], user ratings [13], and hybrid ranking based on several factors, such as frequency of search terms, where in the metadata the search results appear, and the type of the ontology [14].

A number of the studies of the methods for ontology search and ranking conducted some user evaluations. However, to the best of our knowledge, none of these works used the log analysis of user searches to evaluate the ranking. Furthermore, when researchers conducted user studies to evaluate how well the ranking worked (e.g., AKTiveRank [12]), these studies were based on the results from a small number of users. The high number of visitors to BioPortal (more than 100,000 page views and more than 60,000 unique visitors each month) allowed us for the first time to perform an analysis that used thousands of user searches. Thus, both the approach and the scale make our analysis unique.

3 The Framework for Data-Driven Evaluation of Ontology Ranking

The basic idea in our framework is rather simple: when users search a collection of ontologies, our goal is for the user to find what she is looking for in the first result on the page. We use ontology ranking to order the search results and we record in the search log the position of the ontology that the user selected. The more users click on the first result, or the higher the average position that the users click on, the better the ontology ranking that we used to order the results. We explain our framework using the search in BioPortal as an example.

3.1 Ontology Search in BioPortal

BioPortal is a community-based repository of biomedical ontologies [15].[1] At the time of this writing, it contains more than 330 public ontologies with almost six million terms in them. Search across all ontologies is one of the key features of BioPortal. The system indexes all preferred names, synonyms, and property values for all classes across all ontologies. Users search against this index. The users can limit the search only to preferred names or ids of the terms, or choose to include property values. The users can search across all ontologies or in a group of ontologies of interest, or in a single ontology; they can choose to include or to exclude obsolete terms from the search, and so on.

For instance, Figure 1 shows the search results in BioPortal after the user has searched for "myocardial infarction" across all ontologies. The first 22 results correspond to the ontologies that have the exact term "myocardial infarction." We group the result by ontologies. If an ontology has more than one class that

[1] `http://bioportal.bioontology.org`

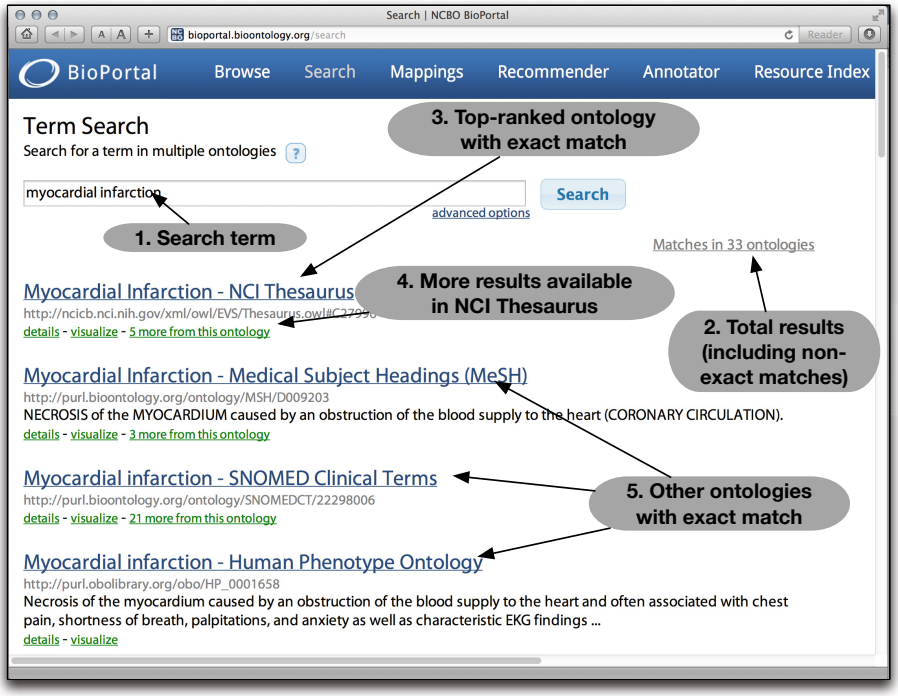


Fig. 1. Search results for "myocardial infarction" in BioPortal: 1. User searches for "myocardial infarction." 2. There are 33 ontologies that contain classes with names, term URIs, or property values that match the search term exactly or partially; of these, 22 ontologies have the exact match. 3. Among the ontologies with the exact match, the NCI Thesaurus has the highest ranking and BioPortal presents it first in the search results. 4. The NCI Thesaurus has 5 more results, which are not necessarily exact matches. 5. The order of other ontologies with exact matches (MeSH, SNOMED CT, etc.) corresponds to their ranking (Table 1, column *Pageviews*).

is relevant to the query, users can access these results by expanding the link for "more from this ontology." For instance, the top result, the NCI Thesaurus, has 12 more search results. The search result shows the pertinent information for the term that matched the user query exactly: the term label, the term URI, and a snippet of a textual definition of the term if the ontology has such definition. The user can also click on a link to have additional details about the term or to have a graph visualizing the neighborhood of the term to appear in a pop-up window. After the user examines the search results, she clicks on the result that seems most relevant to access the term in the ontology browser in BioPortal.

In the example in Figure 1, our search returned 22 ontologies that contain a class with preferred name matching the search string precisely. BioPortal has an ordered ranked list of all its ontologies, which we update regularly. Section 4 discusses the specific ranking approaches that we tested. For instance, the *API+Projects* column in Table 1 shows the top 10 ontologies in the ranking

that BioPortal used when we took the screenshot for Figure 1. In this ranking, among the ontologies that had an exact match for the term "myocardial infarction," the highest rank belonged to NCI Thesaurus. The two ontologies that are ranked higher than NCI Thesaurus (column *API+Projects* in Table 1) do not contain the search term and hence do not appear in the search results.

The rest of the columns in Table 1 present the top 10 ontologies in other ranking orders that we evaluated (Section 4). In order to determine which ranking works better for our users, we recorded user actions in the search logs. Each time a user selects an ontology in the search results to open this ontology in the browser, we record the following data: the search term, the position that the user clicked, whether or not the result was an exact match or an approximate match, the ontologies that were ranked higher than the one that the user selected, the user IP address and other provenance information.

We use the position of the ontology that the user selected as a measure of how effective our ranking was for this particular search. If the user selects the first link and later finds out that this link is not what she was looking for, she will come back to the search results and follow a different link. We record both actions as two different searches.

In order to analyze the effectiveness of a specific ranking relative to another ranking, we compare the collection of positions of ontologies that the users select. We can compare the median and the mean of the position in a set of user search logs. The closer both numbers are to 1 (the user selecting only the highest ranked result), the closer our ranking is to a perfect one.

This framework provides a data-driven evaluation approach to ontology ranking. By varying the internal ranking R, we can compare the effect of various features in composing the ranking: given two rankings, R_i and R_j, the one with the lower mean and median of the positions of selected ontologies is the closer one to a perfect ranking.

3.2 Defining the Data-Driven Evaluation Framework

More formally, consider an ontology collection C and a set of ontologies $\{O_1, O_2, ...O_n\}$ in the collection C. We define a ranking R as a complete order on the set $\{O_1, O_2, ...O_n\}$. When a user searches the collection C for a term t (e.g., "myocardial infarction"), let the set C_t be the subset of ontologies from C that is returned as the result of the search for the term t. In the search results presented to the user, the ontologies in the set C_t are ordered according to the ranking R.

We define the **effectiveness of the ranking** R based on the user behavior after the search engine presents the ontologies in the set C_t ranked according to R. The ranking R is a **perfect ranking** if every user selects the first choice presented by the search engine. The closer the user behavior is to the perfect ranking, the more effective the ranking R is.

Table 1. The top 10 ontologies in each of the four ontology rankings that we used in the study. This ranking dictates the order of search results. The table presents four rankings: The first group are the top 10 ontologies based on pageviews in BioPortal; the second group presents the ranking based on combination of pageviews in BioPortal and API calls; the third group is the ranking based on API calls and use in projects submitted by users; the final group presents the ranking based on combination of all features. See Section 4 for details of the ranking features in.

Pageviews	Pageviews + API
1. National Drug File	1. SNOMED Clinical Terms
2. SNOMED Clinical Terms	2. NCI Thesaurus
3. MedDRA	3. Human disease ontology
4. International Classification of Diseases	4. MedDRA
5. NCI Thesaurus	5. International Classification of Diseases
6. Mouse adult gross anatomy	6. National Drug File
7. RadLex	7. Ontology for Biomedical Investigations
8. Bioinformatics operations... (EDAM)	8. Human Phenotype Ontology
9. Human disease ontology	9. Experimental Factor Ontology
10. RxNORM	10. Medical Subject Headings (MeSH)
API + Projects	**All**
1. Gene Ontology	1. NCI Thesaurus
2. Gene Ontology Extension	2. SNOMED Clinical Terms
3. NCI Thesaurus	3. Ontology for Biomedical Investigations
4. Medical Subject Headings (MeSH)	4. Human disease ontology
5. Ontology for Biomedical Investigations	5. RadLex
6. Foundational Model of Anatomy	6. Experimental Factor Ontology
7. SNOMED Clinical Terms	7. Medical Subject Headings (MeSH)
8. NCBI organismal classification	8. Foundational Model of Anatomy
9. Chemical entities of biological interest	9. NCBI organismal classification
10. Cell type	10. NIF Standard Ontology

3.3 Analyzing and Comparing Rankings

We use the search-log data to analyze the effectiveness of a specific ontology ranking and to compare the effectiveness of different rankings to one another. For our analysis, we use only the results that had the exact match for the search term—these results constitute the first batch of search results that BioPortal presents to users and it orders this set based on its current internal ontology ranking R. For each result, we take the position of the ontology that the user selected. For example, consider five entries in our search log for a period of time when a ranking R_i was active: Suppose one entry indicates that the user selected the ontology in position 2, another entry has the user selecting the ontology in position 10 for her search, and the three remaining entries have the users select the top link. Then, $P_{R_i} = \{2, 10, 1, 1, 1\}$. Thus, we get a set P_R of all positions of ontologies that users have selected over a period of time when the ranking R was active. We analyze the set P_{R_i} for each ranking R_i that we want to evaluate.

In order to analyze each individual ranking R_i, we compute the following metrics for the corresponding set P_{R_i}:

Median Selected Position: the median position that the user selects;
Mean Selected Position: the average value for the position of the ontology that users select; the closer this value is to 1, the closer our ranking is to a perfect ranking for ontology search.
Percentage of Selections in the Top Position: the fraction of users that have selected the top link among the results that the search engine presented.

We use a randomly generated ranking of ontologies R_{random} as a baseline. Presenting ontologies in a random order for several days allowed us to obtain the baseline for user behavior. We created this baseline in order to answer the question of how much the users tend to select the first result that we present, regardless of the ontology rank.

To compare rankings among one another, we performed a series of pair-wise statistical tests based on the Wilcoxon rank-sum test, followed by a Bonferroni correction to reduce the chance of type-I errors due to multiple comparisons. We first perform a one-sided Wilcoxon rank-sum test to determine whether each of the rankings R_i provides a statistically significant improvement over the randomly generated ranking R_{random} as determined by the two corresponding sets of selected ontology positions P_{R_i} and $P_{R_{random}}$ (Test 1). Here, the null hypothesis (H_0) is that the distributions of P_{R_i} and $P_{R_{random}}$ are identical. The alternative hypothesis (H_a) is that the distribution of $P_{R_{random}}$ is shifted to the right of P_{R_i}; in other words, ranking R_i is more effective than R_{random}. A small p-value in this case is an indicator that the location shift (i.e, ranking improvement) is unlikely to due to chance. We then compare each pair of rankings R_i , R_j to each other using a two-sided Wilcoxon rank-sum test to determine whether they are statistically different (Test 2). In this test, H_a is the hypothesis that the distributions of P_{R_i}, P_{R_j} are not identical (location shift is not equal to zero); or in other words, the distributions P_{R_i}, P_{R_j} are statistically different.

In the rest of this paper, we describe the application of this framework to analyze a number of ontology ranking features in BioPortal.

4 Features in BioPortal Ontology Ranking

We have actively solicited suggestions from our user community on what features to use in ranking BioPortal ontologies. As the result of these discussion, we selected the following list of features that could affect the ranking of ontologies:

Pageviews (PV): We use Google Analytics to measure the number of pageviews that each ontology in BioPortal receives. Because BioPortal allows users to browse multiple versions of the same ontology, we aggregate browsing history across versions: whichever version of an ontology O_{V_i} a user

browses, those pageviews contribute to the browsing activity for the ontology O. We use an interval of one month each time to create a new ranking of BioPortal ontologies based on pageviews. This feature measures how frequently users browse an ontology in BioPortal: the more frequently the users browse a particular ontology, the higher its rank.

API Activity (API): Many developers use the NCBO Web services API [15] to access the ontologies from within their applications. Web service calls allow the caller to specify which ontology to use. For example, a group focusing on diseases may use all disease ontologies or specify only the ontologies that they consider to be the "best." The more frequently an ontology is explicitly specified in the Web service API calls, the higher its ranking along this feature. Specifically, we count the number of unique API keys (users) that access each ontology through the API.

Projects (Pr): BioPortal users can describe their ontology-related projects on the BioPortal site. The users can then link these project descriptions to the ontologies that they use in the projects. The more projects use an ontology, the higher its rank based on this feature.

Notes and Reviews (NR): BioPortal users can also provide reviews of ontologies and attach comments (notes) and new term requests to individual classes in an ontology. This activity is another indicator that we take into account to determine the ontology rank.

We ranked the ontologies based on each feature and then combined the *ranks* to create the ranking that relied on more than one feature. We could also add a weight to any of the features if we want to emphasize any one of them. In our experiments to date, we assigned each feature the same weight. We discuss additional features that we can include in ontology ranking in Section 6.

In our experiment, we evaluated the following ontology rankings, with each ranking being active for a period of time. For rankings that use multiple features, we added the ranks for each feature and based the combined ranking on this sum.

Random (R_{random}): provides a baseline for the user search behavior

Browsing Activity only (R_{PV}): reflects the interaction with BioPortal ontologies through the browser

Browsing Activity and API Activity (R_{PV+API}): reflects the general use of an ontology, through the pageviews or through API calls

API Activity and Number of Projects (R_{API+Pr}): reflects the use of the ontology in projects through measuring the explicit links between ontologies and projects as specified by the users on the BioPortal site and the use of the ontology in the API calls that developers make.

All of the Above (R_{All}): reflects a combination of all the measures that we studied. Specifically, it combines PV, API, and Pr, all with equal weights.

Using projects (R_{Pr}) or Notes and reviews (R_{NR}) alone did not differentiate the ontologies significantly, with 87% of the ontologies having at most one note or review. For R_{Pr}, 78% of ontologies had 4 or fewer projects. Thus, we did not yet use this feature by itself for the ranking in the live system. In future work,

we plan to consider additional combination of features that take into account the features with low degree of differentiation, such as R_{NR}. Because there was some variability in the number of projects, with 22 different ranks, we used R_{Pr} in combination with R_{API}.

5 Results

We collected the data on user activity in BioPortal between January 1, 2013 and April 10, 2013 (Table 2).[2] The number of searches for each ranking ranged between 500 and 694. We considered only the searches where the user clicked on one of the ontologies with the exact match. This search behavior was affected the most by the rankings.

We describe the analysis of the features that we used for ranking (Section 5.1), search-log data in Section 5.2 and we compare the effects of features that we described in Section 4 on the effectiveness of ranking in Section 5.3.

5.1 Analysis of the Features

Figure 2 presents the ranges for the features that we considered for the ranking. Recall that when computing combined rank, we used the rank of ontologies for each feature rather than the absolute values for the features. The graphs show that the notes provided too little differentiation between ontologies and thus we did not use them in these experiments.

5.2 Analysis of the Search Data

In the period that we studied, the users performed the total of 4,859 searches. Of these searches, we analyzed the 3,029 searches (62%) where the user selected one of the ontologies with an exact match for the search term. These searches came from 969 unique IP address.

The users searched for 2,276 unique terms. In other words, more than 75% of the search terms appeared only once in searches over a period of 81 days.

The average number of ontologies that BioPortal returned for the searches in our analysis was 11 ontologies with exact matches for the user's search term.

BioPortal users can create an account on the site and log in to the site as they browse. Being logged in allows users, for example, to custom-tailor the set of ontologies that they see (e.g., by limiting this set only to the ontologies that they are interested in), to add reviews and comments on the ontologies, and to describe their projects. We found that only 3% of the searches were performed by users who were logged in to BioPortal during the search.

[2] The exact date when we pushed each new ranking to the BioPortal depended on the release schedule and other operational requirements, resulting in the slight variation in the number of days for each ranking. We decided to keep all the data rather than to truncate each period to 15 days in order to analyze as much data as possible.

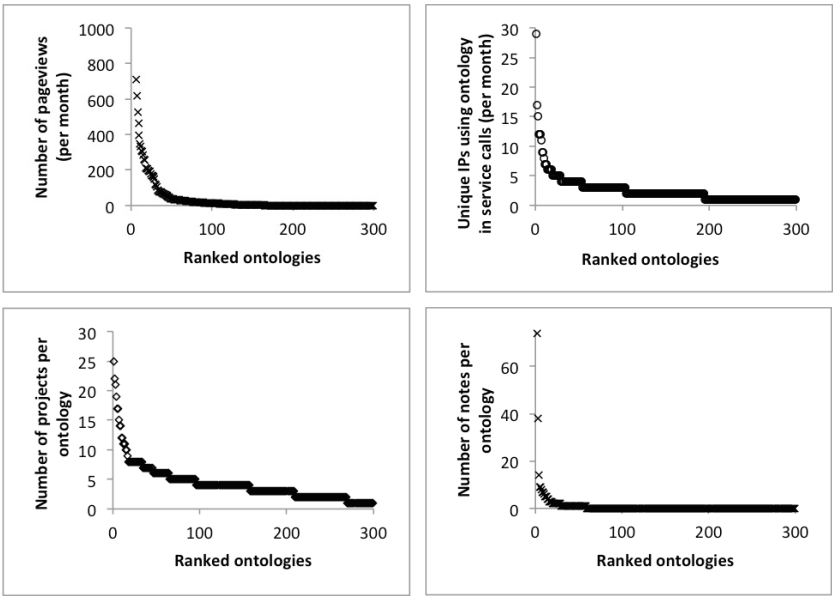


Fig. 2. The distribution of absolute values for the features. The pageviews provide the most discrimination among ontologies, whereas notes and reviews provide essentially none, with most ontologies having fewer than 2 notes. We used the relative rank of an ontology based on the specific feature rather than absolute values. The pageview plot excludes the top 5 ontologies; the monthly pageviews for these ontologies ranged from 1,000 to 10,000.

5.3 Comparing the Rankings

In each ranking that we considered, including the case when we ranked the ontologies randomly, the median position of the selected ontology was 1. In other words, more than half the time, users click on the first search result.

Table 3 displays p-values for Test 1 (Section 3.3), which we used to determine whether each of the four rankings (R_{PV}, R_{PV+API}, R_{API+Pr}, and R_{All}) provides a statistically significant improvement over a randomly generated ranking (R_{random}). According to the information in Table 3, there is strong statistical evidence (extremely small p-values) that the ranking improvement provided by the R_{PV} and R_{PV+API} ranking algorithms is unlikely due to chance (non-random). Furthermore, the p-values support the finding that the R_{PV} and R_{PV+API} algorithms provide performance that is superior to the other ranking algorithms. In other words, using pageviews or pageviews in combination with the API calls as the basis for ranking provides greater improvement compared to using API and projects (R_{API+Pr}) or the combination of all the features (R_{All}), which do not provide performance that is drastically different from the randomly generated ranking. Indeed, as Table 2 shows, the number of searches where the user select the ontology in the top position is 27% and 26% higher than random for R_{PV} and R_{PV+API}, respectively. For R_{PV}, almost 75% of searches result in the

Table 2. The summary information about the ranking algorithms used in the study

	Random	Pageviews	Pageviews +API	API + Projects	All
Period (all dates in 2013)	1/15-1/30	1/1-1/15	3/6-3/21	3/21-4/10	2/4-2/19
Number of days	16	15	15	20	15
Number of searches	500	589	694	639	607
Unique IP addresses	190	168	213	218	180
Searches by logged in users	4	13	11	29	43
Unique search terms	380	455	556	491	490
Unique search terms (%)	76.0%	77.2%	80.1%	76.8%	80.7%
Mean position selected	2.44	1.72	1.78	2.1	2.25
Users selecting top ontology	57.6%	74.4%	72.9%	63.9%	60.8%
Median position selected	1	1	1	1	1

Table 3. Comparing rankings to the random ranking. The p-values to test if improvement in ranking is due to chance (Test 1). The rankings that use Pageviews (R_{PV}) and Pageviews with API (R_{PV+API}) provide performance that is statistically significant.

	Pageviews	Pageviews + API	Projects+API	All
Random	1.26E-09	1.64E-09	0.01192	0.3306

selection of the top link. Notwithstanding, when comparing R_{PV} and R_{PV+API} to each other (Test 2, Section 3.3) we find that the two rankings are statistically indistinguishable from each other (p-value=0.67). This data suggests that combining the *API* feature with the *PV* feature does not provide a significant performance improvement over using the *PV* feature by itself.

6 Discussion

In this paper, we have developed a framework that enables us to evaluate ontology ranking algorithms in a data-driven way. Indeed, we need only to swap out one ranking for another and to continue to collect the data in order to compare different ranking. Because of the relatively high volume of searches on BioPortal, we get sufficient data to determine whether or not a ranking algorithm is working in a matter of a couple of weeks.

6.1 Changes in Ontology Ranking

We start our discussion by providing a sense of how much movement we observed in the four rankings of BioPortal ontologies that we presented in this study. There are more than 330 ontologies in BioPortal and their order differed significantly

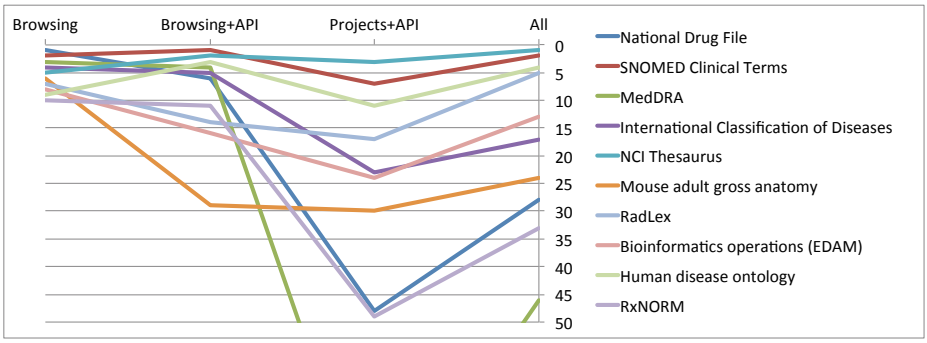


Fig. 3. Rank changes for the top 10 ontologies in the R_{PV} ranking (the ranking based on pageviews). Each line indicates the rank of the ontology based on the corresponding ranking algorithm. The graph captures the ranks between 1 and 50. The line for MedDRA (green) drops off the chart for the ranking based on Projects and API because MedDRA was ranked 89 in that ranking.

from one ranking to another. We compare the movement of ontologies in the rankings relative to the R_{PV} ranking, the ranking that performed the best in our evaluation. Consider the graph in Figure 3, which tracks the ranks of the top ten ontologies in the R_{PV} ranking. Each line represents the rank for a single ontology among these top ten, when we use the corresponding features for the ranking. The ranks for these ontologies ranged from 1 to 89 in the other rankings. We observed the biggest shift from the R_{PV} ranking in the R_{All} ranking, a ranking based on combination of all features. Indeed, the MedDRA terminology, which is ranked first based on page views, was ranked 89th in the ranking based on projects and APIs—an indication that while users often browse MedDRA in BioPortal, they do not use it in their ontology-related projects or access it through the BioPortal API.

Table 4 shows the average number of positions that the ontologies moved up or down relative to the R_{PV} ranking, for the top 100 ontologies. On average, each ontology that moved higher in the ranking, compared to R_{PV}, moved by 17.3 spots in the ranking. Each ontology that moved down in the ranking, moved by 60.7 spots, with the largest average movement between the ranking based on projects and API, R_{Pr+API}, and the ranking based on pageviews, R_{PV}. This result is not surprising because R_{Pr+API} is the only ranking among the ones that we considered that does not take pageviews into account.

6.2 Comparing the Rankings

Our analysis of the four ranking approaches for BioPortal ontologies demonstrated several trends. First, the majority of users select the top link, regardless of the ontology that it comes from. This observation is similar to the results that Joachims and colleagues [16] reported for regular Web search and what they referred to as "Trust bias." The fact that the user behavior changes as the

Table 4. The average movement distance (in the position change) for ontologies relative to the R_{PV} ranking. The data is for the top 100 ontologies in the R_{PV} ranking.

	Pageviews + API	Projects + API	All
Moving *higher* in the ranking	12	24	16
Moving *lower* in the ranking	-52	-73	-57

ranking changes confirms the "quality bias" reported by Joachims and colleagues: the quality of the ranking does affect the clicking behavior of the users. The trust bias appears to be more pronounced in ontology search than in regular web search, possibly because it is harder for users to assess the quality of the result from the snippets that BioPortal provides. For example, not all terms in ontologies have textual definitions, and therefore, the only information that the user might see is the term name and id. This information may not be enough to make informed decision.

Therefore, the better we are at putting the most relevant ontology at the top of the list, the more satisfied the users will be. Second, the rankings that performed the best in our experiments, R_{PV} and R_{PV+API}, were the ones that reflected the activity of users in the BioPortal user interface. In both rankings, the analysis of pageviews for an ontology played the key (or the only) role. This result is not surprising: indeed, the users who interact with the BioPortal search interface—the ones whose logs we used in the analysis—are exactly the users who browse BioPortal. The other rankings had a stronger component from the developers and users who already know which ontologies they need and thus were less helpful in ranking the ontologies in the user interface. These rankings did not improve the effectiveness of the search.

6.3 Other Condiserations

In our study, we focused on the users who perform ontology search. On the one hand, such filtering allowed us to rely on a smaller number of users who perform the same task [17]. At the same time, this decision led to several limitations.

First, if a user selected the top ontology, was not satisfied and then came back and selected a lower ranked one, we will record both selections in the log. This analysis is equivalent to the "click > skip above" strategy described Joachims and colleagues [16]. That work demonstrated that this strategy of assuming that the user finds any clicked result more relevant than the results above it, provide to be one of the most accurate strategies.

In reality, the user did not find what she was looking for in the ontologies that she selected first. Indeed, many users may not have precise or explicit criteria to select the ontology that will satisfy their needs and many of the searches might be exploratory. In order to be more precise about the satisfaction of the user, we may want to count only the last of the positions in a batch of selections from the same IP address with the same search term. Our initial analysis of the

data indicates that this change will not have a significant effect on the results because the search logs are dominated by unique search terms. However, we plan to perform the detailed analysis that takes into account the history of consecutive selections from the same user.

Second, we are of course unlikely to have a ranking where every user will select the first search result because users have different requirements and might be interested in different ontologies. The best we can do is get the best result as the top result for as many users as possible. We could also use the user personal preferences and search history to custom-tailor the order. For instance, we can monitor the user's behavior and the ontologies that the specific user browses more frequently, and rank those ontologies higher for the specific user. Recall, however, that only 3% of the searches in our study came from the users who were logged in and "known" to the system.

Furthermore, we currently do not take the search results within the ontology into account: whether an ontology has several non-exact hits on the search term or only one does not effect its ranking for the specific search result. In the future, we can add this information to the ranking for a specific search.

We do not normalize pageviews–the key indicator in the ranking–by the ontology size, a decision that maybe counter-intuitive at first glance. However, it generally takes as much time on behalf of the user to perform X pageviews in a large ontology as it does in a small ontology. Because each page view corresponds to an explicit action by a user, this metric does not privilege large ontologies. However, because large ontologies have broader coverage and are more likely to appear in search results, uses might visit them more often for that reason. Large ontologies (e.g., SNOMED CT, ICD) also usually have some institutional support behind them and thus users are more likely to use those ontologies.

Finally, the ranking that we produce is only as good as the information that we use as input to the ranking. For instance, we believe that the project information is incomplete as many BioPortal users have not entered information for their projects. We are involved in an active outreach effort to expand the coverage of project descriptions. As these descriptions become more comprehensive, the effect of this feature on the ranking may change as well. Similarly, we we get more notes and reviews on the ontologies, that feature will differentiate the projects more and will have a different effect on the ranking. We plan to use our framework to re-evaluate the effects of these features continuously.

6.4 Future Work

Our analysis points to several future directions in improving ontology ranking methods—methods that we can continue testing in our framework. First, we can consider different weights on the features that go into the ranking. For example, we can weigh the rank based on pageview more, but still include other features. Second, we can use our framework to investigate a number of other features that can contribute to ontology ranking, in addition to the features that we have described in this paper. For example, we can consider the following features:

- the percentage of ontology terms that have textual definitions: if ontology developers took care of providing natural-language description for all, or most, of the terms, it might indicate that ontology is more useful for users;
- the number of other ontologies that import an ontology or reuse its terms: if an ontology is frequently reused, it might be ranked higher than others;
- coverage of a document corpus: we use ontologies to index records in many public datasets; an ontology where higher percentage of the terms that are reflected in large teal-life corpora may be more useful.

Our framework enables us to evaluate the effectiveness of ontology ranking for the purposes of ontology search. These result do not necessarily translate to a more general solution to ontology-evaluation. Indeed, as many researchers have pointed out, the best way to approach ontology evaluation is through task-specific evaluation [18]. While there is likely a correlation between the ranking for the purposes of improving the user search experience and more general ontology evaluation, we need to investigate this link in further research.

Note that these and other features and their positive or negative effect on ontology ranking are the hypotheses that we can test in our framework. Our results so far have demonstrated that some "common-sense" hypotheses do not necessarily hold if we analyze search data.

In our experiments, we focused exclusively on the search task. Analyzing the user behavior throughout the system, including their browsing of ontologies, will give us a more complete picture of user satisfaction. For example, the usage logs can reveal whether users explore multiple ontologies before settling on a single one. We can analyze how much time users spend on each ontology, how much time they spend on the pages following the search, and what actions they perform. Analyzing the data beyond the search page will give us a more complete picture of the user behavior and their implicit satisfaction with the search results.

7 Conclusions

Our framework provides an efficient way to compare various approaches to ontology ranking in a data-driven way by analyzing the user behavior in selecting search results. Our analysis of different ranking approaches for biomedical ontologies in BioPortal, shows that the majority of users always select the first search result, making good ontology ranking ever more important for user satisfaction.

Acknowledgements. This work was supported by the National Center for Biomedical Ontology, under grant U54 HG004028 from the National Institutes of Health. We are indebted to Barry Smith for encouraging us to look critically at the ranking of search results. We are grateful to our NCBO collaborators Helen Parkinson, Simon Jupp, and James Malone for their suggestions on features to use in ontology ranking.

References

1. d'Aquin, M., Noy, N.F.: Where to publish and find ontologies? A survey of ontology libraries. Journal of Web Semantics (JWS) 11, 96–111 (2011)
2. Musen, M.A., Noy, N.F., Shah, N.H., Whetzel, P.L., Chute, C.G., Storey, M.A., Smith, B.: The NCBO team: The National Center for Biomedical Ontology. Journal of American Medical Informatics Association 19, 190–195 (2012)
3. Pan, J.Z., Thomas, E., Sleeman, D.: Ontosearch2: Searching and querying web ontologies. In: IADIS International Conference WWW/Internet, pp. 211–219 (2006)
4. Buitelaar, P., Eigner, T., Declerck, T.: OntoSelect: A dynamic ontology library with support for ontology selection. In: Demo Session at the International Semantic Web Conference (ISWC 2004) (2004)
5. Alani, H., Noy, N.F., Shah, N.H., Shadbolt, N., Musen, M.A.: Searching ontologies based on content: experiments in the biomedical domain. In: 4th Int. Conf. on Knowledge capture (K-CAP 2007), pp. 55–62. KCAP, Whistler (2007)
6. Sabou, M., Lopez, V., Motta, E.: Ontology selection on the real semantic web: How to cover the queens birthday dinner? In: 15th Int. Conference on Knowledge Engineering and Knowledge Management (EKAW), Czech Republic (2006)
7. Resnik, P.: Semantic similarity in a taxonomy: An information-based measure and its application to problems of ambiguity in natural language. Journal of Artificial Intelligence Research 11, 95–130 (1999)
8. Voorhees, E.M.: Query expansion using lexical-semantic relations. In: SIGIR 1994: 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 61–69. Springer-Verlag New York, Inc. (1994)
9. Salton, G., McGill, M.J.: Introduction to Modern Information Retrieval. McGraw-Hill (1983)
10. Subhashini, R., Akilandeswari, J., Sinthuja, V.: Article: A review on ontology ranking algorithms. International Journal of Computer Applications 33(4), 6–11 (2011); Published by Foundation of Computer Science, New York, USA
11. Ding, L., et al.: Swoogle: A search and metadata engine for the semantic web. In: Conf. on Information and Knowledge Management (CIKM), Washington (2004)
12. Alani, H., Brewster, C., Shadbolt, N.R.: Ranking ontologies with AKTiveRank. In: Cruz, I., Decker, S., Allemang, D., Preist, C., Schwabe, D., Mika, P., Uschold, M., Aroyo, L.M. (eds.) ISWC 2006. LNCS, vol. 4273, pp. 1–15. Springer, Heidelberg (2006)
13. d'Aquin, M., Lewen, H.: Cupboard–a place to expose your ontologies to applications and the community. In: Aroyo, L., Traverso, P., Ciravegna, F., Cimiano, P., Heath, T., Hyvönen, E., Mizoguchi, R., Oren, E., Sabou, M., Simperl, E. (eds.) ESWC 2009. LNCS, vol. 5554, pp. 913–918. Springer, Heidelberg (2009)
14. Jonquet, C., Musen, M., Shah, N.H.: Building a biomedical ontology recommender web service. Journal of Biomedical Semantics 1(suppl. 1), S1 (2010)
15. Whetzel, P.L., Noy, N.F., Shah, N.H., Alexander, P.R., Nyulas, C.I., Tudorache, T., Musen, M.A.: BioPortal: Enhanced functionality via new web services. Nucleic Acids Research (NAR) 39(Web Server issue), W541–W545 (2011)
16. Joachims, T., Granka, L., Pan, B., Hembrooke, H., Gay, G.: Accurately interpreting clickthrough data as implicit feedback. In: 28th Annual International ACM SIGIR Conference, pp. 154–161. ACM, Salvador (2005)
17. Kohavi, R., Henne, R.M., Sommerfield, D.: Practical guide to controlled experiments on the web: listen to your customers not to the HiPPO. In: 13th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, San Jose, CA (2007)
18. Hoehndorf, R., Dumontier, M., Gkoutos, G.V.: Evaluation of research in biomedical ontologies. Briefings in Bioinformatics (2012)

Exploring Scholarly Data with Rexplore

Francesco Osborne[1,2], Enrico Motta[1], and Paul Mulholland[1]

[1] Knowledge Media Institute, The Open University, MK7 6AA, Milton Keynes, UK
{francesco.osborne,enrico.motta,paul.mulholland}@open.ac.uk
[2] Dept. of Computer Science, University of Torino, 10149 Torino, Italy
osborne@di.unito.it

Abstract. Despite the large number and variety of tools and services available today for exploring scholarly data, current support is still very limited in the context of sensemaking tasks, which go beyond standard search and ranking of authors and publications, and focus instead on i) understanding the dynamics of research areas, ii) relating authors 'semantically' (e.g., in terms of common interests or shared academic trajectories), or iii) performing fine-grained academic expert search along multiple dimensions. To address this gap we have developed a novel tool, Rexplore, which integrates statistical analysis, semantic technologies, and visual analytics to provide effective support for exploring and making sense of scholarly data. Here, we describe the main innovative elements of the tool and we present the results from a task-centric empirical evaluation, which shows that Rexplore is highly effective at providing support for the aforementioned sensemaking tasks. In addition, these results are robust both with respect to the background of the users (i.e., expert analysts vs. 'ordinary' users) and also with respect to whether the tasks are selected by the evaluators or proposed by the users themselves.

Keywords: Scholarly Data, Visual Analytics, Data Exploration, Empirical Evaluation, Ontology Population, Data Mining, Data Integration.

1 Introduction

Understanding what goes on in a research area is no easy task. Typically, for a given topic, this *sensemaking process* may require exploring information about a variety of entities, such as publications, publication venues, researchers, research groups, events, and others, as well as understanding the relationships which exist between them. Such exploration and sensemaking tasks can take place in a variety of contexts, involving different categories of users. For instance, one of the authors of this paper is Editor-in-Chief of a scientific journal and in such a role he regularly needs to consider competing proposals for special issues, a task which requires (among other things) to analyze the dynamics of one or multiple research areas, in order to formulate a view on whether the proposals in question concern areas that are 'hot' and growing, or are instead to a lesser extent at the cutting edge. In other task contexts, such *scholarly data* are also of great interest to research managers, funding bodies and government agencies, who i) may want to find out about the performance of specific individuals and groups, and compare them with their peers both at national and international level; or ii) may need to gather objective evidence about research trends to inform funding policy decisions.

H. Alani et al. (Eds.): ISWC 2013, Part I, LNCS 8218, pp. 460–477, 2013.

Obviously, there are many tools and services currently available, which already provide a wide variety of functionalities to support the exploration of scholarly data – see Section 2.1 for a review of the state of the art. Nevertheless, as Dunne et al. point out [1], there is still a need for an *integrated solution*, where the different scholarly tasks are provided in a coherent manner, through an environment able to support a seamless navigation between different views and functionalities. In addition, as discussed in detail in the next section, we believe that there are also a number of important functionalities, which are crucial to providing effective support for exploring and making sense of scholarly data, but are currently missing from existing solutions. These include (but are not limited to) the ability i) to investigate research trends effectively at different levels of granularity, ii) to relate authors 'semantically' (e.g., in terms of common interests or shared academic trajectories), and iii) to perform fine-grained academic expert search along multiple dimensions.

To address this gap we have developed a novel tool, Rexplore [2], which integrates statistical analysis, semantic technologies, and visual analytics to provide effective support for exploring and making sense of scholarly data. In this paper, we illustrate the main innovative elements of the tool and we also present the results from a task-centric empirical evaluation, which shows that Rexplore is highly effective at providing support for the aforementioned sensemaking tasks. In addition, these results are robust both with respect to the background of the users (i.e., expert analysts vs. 'ordinary' users) and also with respect to whether the tasks are selected by the evaluators or proposed by the users themselves.

2 Exploring Scholarly Data

2.1 State of the Art

A large variety of systems support the exploration of scholarly data, some of them providing an interface to a specific repository of bibliographic data, others integrating multiple data sources to provide access to a richer set of data and/or to provide a richer set of functionalities. The most widely used academic search engine is probably Google Scholar (http://scholar.google.com), which primarily supports search and citation services, providing comprehensive access to the academic literature. DBLP (http://www.informatik.uni-trier.de/~ley/db/) is a well-known computer science bibliography website and can be browsed using FacetedDBLP [3], an interface which exploits the faceted search paradigm to support data exploration. CiteSeer^X [4] focuses instead on large-scale harvesting and indexing of research papers and includes mechanisms for suggesting relevant papers. These systems mainly focus on providing a good interface for publication search and are not designed to support sensemaking tasks in the academic domain. On the contrary, Microsoft Academic Search (http://academic.research.microsoft.com/) provides a variety of visualizations, including co-authorship graphs, publication trends, and co-authorship paths between authors. In a similar way Arnetminer [5] also offers different visualizations and provides support for expert search and trend analysis. Saffron [6], which builds on the Semantic Web Dog Food Corpus [7], exploits keywords for expert search and estimates the strength of an author/topic relationship by analyzing co-occurrences on the Web. A common aspect of these systems is that they use keywords extracted from

publications as proxies for research topics. However these are noisy and lack structure (see Section 2.2.1 for a detailed discussion on this aspect).

Recently, reference management tools have emerged, such as Zotero (http://www.zotero.org), EndNote (http://endnote.com) and Mendeley (http://www.mendeley.com), as well as specialized social networks sites for researchers –e.g., ResearchGate (http://www.researchgate.net) and Accademia.edu (http://www.academia.edu). However, while these systems support exploration to some degree, again they only provide limited support for sensemaking tasks.

A key challenge for a system exploring scholarly data is how to assist users in searching and navigating through a variety of different dimensions –e.g., topic, organization, co-author, etc. A popular paradigm is *faceted browsing* [8], in which a set of objects can be filtered progressively along several dimensions in different orders. The \facet tool [9] exploits this idea to allow for an easier exploration of heterogeneous Semantic Web repositories by using the different resources found in RDF repositories as alternative facets. mSpace [10] tackles the problem of dealing with high-dimensional spaces, by showing a subset of the data at the time, called "a slice", and arranging them in a hierarchy of columns in accordance with user-defined priorities. Other approaches rely on the *pivot* (or multi-pivot) paradigm [11], which allows users to identify key elements in the data space (the pivots), and use these to introduce structure and facilitate the navigation process. For example, PaperCUBE [12] offers advanced data visualization functionalities and it specifically focuses on scholarly data, providing effective visual modalities to browse citation networks and relations between authors and to situate a paper in a research context. However, the focus here is primarily on individual publications and little support is provided for higher-level tasks, such as understanding research dynamics and fine-grained expert search.

2.2 Gap Analysis

As we have seen in the previous section, the space of solutions for exploring scholarly data is large, comprising both powerful systems for crawling and indexing scholarly data, such as Google Scholar, as well as a variety of visualization solutions and data exploration paradigms, some generic in nature, others specifically customized for scholarly data. However, despite the availability of such a variety of systems, exploring scholarly data remains challenging, especially once we move away from basic search (for authors or publications) and we aim to capture the dynamic elements to do with research trends and relationships between authors (which go beyond citation and collaboration), or we aim to perform expert search at a very fine-grained level –e.g., by searching for researchers with expertise in multiple topics, at a certain career stage, within a certain geographical area, who have a track record of publishing in the top conferences associated with one or multiple research areas, etc. In what follows we will discuss these issues in more detail, highlighting the key gaps that Rexplore aims to address.

2.2.1 No Semantic Characterization of Research Areas

A key precondition for an effective exploration of scholarly data concerns the mapping of people and publications to the relevant research areas. However, 'research

area' is rarely treated as a first class concept and instead systems tend to use keywords as proxies for research areas. This limitation creates a number of problems. For instance, the Arnetminer page for Enrico Motta includes "International Semantic Web Conference" as a research interest, even though research interests should arguably concern topics[1], rather than conferences. A similar problem can be seen by looking at the Microsoft Academic Search (MAS) page for Enrico Motta, which lists three high level 'fields' for him, "Database", "Web", and "Artificial Intelligence", and then supplements this information with a number of keywords, including "Case Study", which (again) is arguably not a research area.

Another problem stemming from a syntactic, rather than semantic, treatment of research areas is that systems do not take into account important semantic relations between research areas, such as an area being a sub-area of another one, or two labels referring to the same research area. This problem has been traditionally addressed by relying on manually curated taxonomies, such as the ACM classification (http://www.acm.org/about/class/). However these classifications suffer from several problems. First of all, they are very shallow –for example the entry "Intelligent Web Services and Semantic Web" in the ACM classification only contains four sub-topics, thus failing to reflect the variety of topics being tackled by the Semantic Web research community. In addition, because they are manually curated, they evolve very slowly and as a result, they fail to reflect the latest research trends. Finally, they are actually very opaque, as it is not clear what does it mean for a topic to be classified under another topic. For instance, "Ontology Languages" is classified under "Intelligent Web Services and Semantic Web"; however one could argue that it is strange to say that the former is a sub-topic of the latter, given that ontology languages were being designed well before the Semantic Web was recognized as a research area. In addition, these classifications do not cater for situations where there are different ways to refer to the same area. For instance, most people would agree that the labels "Ontology Matching" and "Ontology Alignment" refer to the same area of research.

2.2.2 Lack of Granular Analysis

Systems such as MAS provide ways to visualize research trends. However, these are considered at a very high-level of abstraction. For example, MAS can visualize publication trends in "World-Wide-Web" and "Databases", but cannot provide this feature for "Semantic Web", let alone more fine-grained topics, such as "Semantic Web Services". However, both researchers and students tend to be interested in rather fine-grained trends – e.g., what's happening with Linked Data, rather than what's happening with the Web. A wider range of topics is provided by Arnetminer, however these still cover only a subset of the research topics (e.g., key topics for the Semantic Web community, such as "Linked Data" and "Ontology Evolution" are not included) and in addition they are provided as a flat list, rather than in a structured, easily navigable form.

[1] In what follows, we will use the terms 'topic' and 'research area' interchangeably.

2.2.3 Digital Library Bias

Another limitation of most existing systems in the context of the sensemaking tasks that we wish to support is the emphasis on classic digital library functionalities, such as supporting search for publications and providing citation services. While of course these are key functionalities and essential building blocks for more advanced services, they do not necessarily provide the right level of support when the goal is to make sense of what goes on in a research area, rather than to identify a specific paper. For instance, in the example given in Section 1, where a research area needs to be investigated in the context of making a decision about a special issue proposal, what is needed from a system is the ability to support the user in identifying quickly the important trends in the area –such as, whether it is growing (and in this case where are the new researchers coming from) or shrinking (and in this case where are the researchers migrating to), rather than following citation links or locating a specific paper. Another negative side-effect of this 'bias' is the aforementioned problem highlighted by Dunne et al. [1], concerning the lack of an integrated environment, supporting a seamless exploration of the space of scholarly data, as opposed to providing 'atomic' functionalities, to do with static visualizations or search and citation services, which is the situation with most current systems.

3 Overview of Rexplore

The goal of Rexplore is to provide an environment capable of overcoming the limitations discussed in the previous section to support users effectively by enabling them i) to detect and make sense of the important trends in one or more research areas, ii) to identify researchers and analyze their academic trajectory and performance in one or multiple areas, according to a variety of fine-grained requirements, iii) to discover and explore a variety of dynamic relations between researchers, between topics, and between researchers and topics, and iv) to support ranking of specific sets of authors, generated through multi-dimensional filters, according to various metrics.

Rexplore addresses the problem of the lack of a semantic characterization of research areas by introducing a fine-grained, automatically populated topic ontology, in which topics are identified and structured according to a number of semantic relationships [13]. The resulting knowledge base is generated using a combination of statistical methods and background knowledge on the basis of a large-scale corpus of publications (Section 3.1) and is then augmented with geographic information (Section 3.2). Research topics can then be browsed and analyzed by means of a variety of visual analytics solutions, which exploit the rich set of relations in the data, and in particular the fine-grained characterization of research areas (Section 3.3). Authors can be investigated by plotting a number of metrics on a timeline, and their associated research areas can be analyzed at different levels of abstraction (Section 3.4). Powerful query/search facilities are also provided, supporting complex multi-dimensional queries that can include logical connectives (Section 3.5). Finally, Rexplore also takes advantage of the fine-grained semantic characterization of authors

and topics, to introduce novel relationships between authors, which go beyond co-authorship and focus on their similarity with respect to 'semantic' features, such as research interests and academic trajectories (Section 3.6).

As shown in Figure 1, Rexplore integrates a variety of data sources in different formats, including: DBpedia [14], DBLP++ (http://dblp.l3s.de/dblp.rdf.gz), the MAS API, GeoNames (http://www.geonames.org/), and parsed web pages (e.g., Wikipedia, Google Scholar). The publication and author metadata used in the current version of the system, Rexplore v2.0, come mainly from MAS and DBLP++. The process of generating the populated topic ontology exploits information collected from Google Scholar, EventSeer (http://eventseer.net/) and Wikipedia. The geographic information and the standardization of the affiliations rely on information from DBpedia and GeoNames (see Section 3.1). As of April 2013, Rexplore integrates metadata on 20 million papers and 2 million authors. The back-end of Rexplore is implemented in PHP, while the interface and the visualizations are in HTML5 and JavaScript. The interface uses the Highcharts (http://www.highcharts.com/) library, and also builds on a heavily modified version of Jit (http://philogb.github.io/jit/).

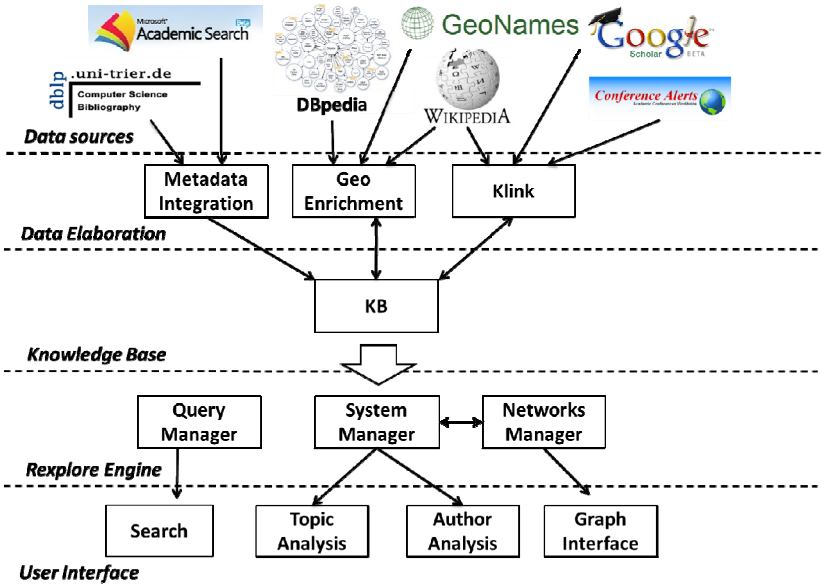


Fig. 1. Rexplore Architecture

3.1 Ontology Population with Klink

Rexplore does not consider topics as simple keywords, but relies on an OWL ontology, which characterizes research areas and their relationships. This ontology is automatically populated and periodically updated by the Klink algorithm [13], which takes as input a corpus of publications, annotated with keywords (these can be user-defined or

automatically extracted from the text of a publication) and performs three key operations, using a combination of statistical methods and background knowledge:

- It identifies research areas from the given set of keywords, tidying them up by fixing errors and by removing keywords that do not denote research areas – e.g., "Case Study" or "NeOn Project".
- It automatically computes three types of semantic relationships between research areas – see below for more details.
- It returns a knowledge base of semantic relationships expressed in OWL.

In particular, Klink computes the following three relationships between topics:

- *skos:broaderGeneric*. This is used to indicate that a topic, say T_1, is a sub-topic of another topic, say T_2. For instance, "Semantic Web Services" can be characterized as a sub-topic of both "Semantic Web" and "Web Services".
- *contributesTo.* This is defined as a sub-property of *skos:related* and it is used to characterize relations where there is evidence (gathered through statistical methods and/or background knowledge) that research in topic T_1 is seen as an important contribution to research in topic T_2, but it would be incorrect to say that T_1 is a sub-topic of T_2. An example is the relation between "Ontology Engineering" and "Semantic Web", where there is significant evidence that results from the former are relevant to the latter, but it would be incorrect to say that "Ontology Engineering" is a sub-topic of "Semantic Web", given that it is a much older research area than "Semantic Web" and, even today, there is a lot of work in Ontology Engineering, which is carried out independently of Semantic Web research.
- *relatedEquivalent*. This is also defined as a sub-property of *skos:related* and it is used to indicate that two keywords, e.g., "Ontology Matching" and "Ontology Alignment" are simply different labels for the same research area[2].

Our ontology[3] builds on the BIBO ontology, which in turn builds on SKOS,[4] FOAF,[5] and other standards. Our extensions are very conservative and comprise only

[2] Here we could have used *Owl:sameAs*, given that Rexplore functionally treats two *relatedEquivalent* topics as being the same one. However, from an epistemological point of view, it can be argued that this would be too strong a commitment and that in other scenarios one may want to consider topics with different names as different ones. Hence, to avoid overcommitting our ontology, we have introduced the *relatedEquivalent* property.

[3] `http://kmi.open.ac.uk/technologies/rexplore/ontologies/BiboExtension.owl`.

[4] The most recent specification of the SKOS model, which can be found at `http://www.w3.org/TR/2009/REC-skos-reference-20090818/`, proposes a new property, *skos:broaderTransitive,* to support the representation of transitive hierarchical relations. However, our ontology currently sticks to the older SKOS specification, primarily because it builds on the BIBO ontology, which in turn builds on the 2004 SKOS model.

[5] `http://xmlns.com/foaf/spec/`

the *relatedEquivalent* and *contributesTo* object properties described earlier, and the class *Topic*, which is used to refer to research topics. The resulting OWL knowledge base is exploited to support knowledge-based exploration, pattern extraction and author clustering in Rexplore. Currently it comprises 1500 topics linked by almost 3000 semantic relationships. A detailed description of Klink, including an empirical evaluation of the algorithm can be found in [13].

3.2 Geographic Enrichment

The data sources used by Rexplore offer in most cases only the name of the author's affiliation (e.g., Universities, Research Labs, Hospitals), which is usually derived from parsing research papers and thus it is simply treated as a string. As a result, affiliations may in some cases lack the actual geographical location or may use different ways to refer to the same institution –e.g., "University of Turin" and "University of Torino". Since a correct affiliation linked to the correct geographic location provides valuable information for filtering and exploring authors, we use a simple but effective geographic enrichment procedure which i) defines a standard name for each affiliation, avoiding duplications, and ii) maps the affiliation to GeoNames, a well-known geographic database. The procedure uses initially Wikipedia to retrieve a 'standard' identifier for the affiliation and then searches for the location associated with the affiliation in DBpedia. If the latter search is unsuccessful, then the Wikipedia page is parsed for the tag "location" from which city and country are extracted using a set of heuristic rules. After recovering information about the city or the country, the affiliation is mapped to the correct GeoNames ID. If the search for affiliation and/or location in Wikipedia/DBpedia fails, then the affiliation name is stripped of a set of typical terms, such as "university", "college" or "hospital", and the remaining string is searched for in the GeoNames database. This simple method provided good results, allowing us to correctly map disambiguated affiliations to GeoNames in about 85% of the cases.

3.3 Topic Analysis

Rexplore takes advantage of the Klink-generated OWL knowledge base by considering every publication tagged with topic T_1 to be also about topic T_2, if T_2 is *broaderGeneric* than T_1, or *relatedEquivalent* to T_1 (it should be noted that *broaderGeneric* is transitive). This has a dramatic effect on the quality and dimension of data available for each topic: for example, our knowledge base includes 11,998 publications tagged with the *string* "Semantic Web", while the publications regarding the *topic* "Semantic Web" (including sub-topics, such as "Linked Data") are almost double (22,143).

Fig. 2. Exploring the topic "Semantic Web" in Rexplore

For analyzing a topic, Rexplore provides an interface that includes: i) general information about the topic, ii) access to the relevant authors and publications, iii) the *topic navigator*, iv) visual analytics on *broaderGeneric* and *contributesTo* sub-topics, and v) visual analytics on authors' migration patterns from other topics to and from the topic in question. As an example, Figure 2 shows the page for the topic "Semantic Web", which (on the left) includes basic statistics, access to basic functionalities, and the topic navigator showing the relevant fragment of the topic hierarchy generated by Klink. On the right hand side of the figure, a histogram is shown, as the user has selected to visualize the publication trends for research in Linked Data, OWL and RDF. In particular, Figure 2 shows that the Linked Data area has exploded in the past few years, while research in OWL appears to have reached a plateau.

Rexplore is able to visualize different topic trends: 1) publication trends, 2) author trends and 3) migration trends. The first two are the number of publications or authors associated with a semantically enriched topic on a timeline. The latter is defined as the number of estimated migrations between two topics and is computed by analyzing the shifting in authors' interest, as described in [15].

3.4 Author Analysis

Every author in Rexplore has a personal page which includes i) general bio information, ii) author's scores according to different bibliometric measures, iii) *topic analysis*, iv) *co-author analysis*, v) *pattern analysis*, and vi) *graph view*. The page offers the possibility of deploying more than 20 different charts to plot each metric as a function of time. The *topic analysis* makes it possible to browse and plot on a timeline the main research areas in which the author has published or was cited. The topics and sub-topics are displayed in a multilevel list in such a way that it is possible to choose the granularity level. For example it is possible to conduct a high

level analysis by focusing on the main topics (e.g., "Semantic Web" or "Artificial Intelligence") or otherwise to zoom in one of them (e.g., "Semantic Web") and further analyze its sub-topics in details, exploiting the semantic structure generated by Klink. The *co-author analysis* section ranks the co-authors according to the number of publications or citations they have in common. It is also possible to select a number of co-authors and visualize their collaboration with the author in question by year and by topic. The *pattern analysis* section groups authors with a similar publications/citations pattern and can be also used to forecast future publication activity and impact for an author (in particular one at a reasonably early career stage). The *graph view* will be discussed in detail in Section 3.6.

3.5 Faceted Search and Data Browsing

Rexplore offers a number of facets to be used both for the formulations of complex search queries and for context-based data navigation and analysis. Indeed, both the topic and author analysis interfaces offer the possibility of focusing on specific combinations of facets, in order to allow the users to navigate/retrieve data according to specific dimensions. For example, authors can be filtered by 1) name or a part of it, 2) career range (that is the time from the first published work), 3) topics of interest and 4) venues in which they published. Both venue and topic fields accept multiple values, which can be combined using logical connectives. Hence it is easy to formulate complex queries, e.g., to retrieve career-young authors, who have worked in both "Semantic Web" and "Social Networks", and have published in ISWC.

The results can be ranked by a variety of metrics that, for author-centric searches, include: 1) number of publications, 2) number of citations, 3) H-Index, 4) G-Index, 5) HT-Index, 6) GT-Index, 7) number of publications/citations in a topic or set of topics, 8) number of publications/citations in a venue or set of venues. Here it is worth to highlight that the fine-grained structure of research topics generated by Klink supports the definition of fine-grained impact metrics, such as "citations in topics", which allow to measure very specific elements of academic impact.

HT-Index and GT-Index are based on the standard G-Index and H-Index, however they are normalized by the number of average citations in each topic. Hence they are useful for comparing authors who publish in fields with different levels of field-specific impact.

Often users want to start the data exploration process from the query results, for example by analyzing each one of a number of authors. Rexplore assists this seamless navigation by remembering the specified search filters –e.g., when switching from a list of results to a graph view.

3.6 The Graph View

The *graph view* is a novel, highly interactive tool to explore the space of authors and their relationships using faceted filters. It takes as input one or multiple authors and

displays their relations allowing the user to choose among a variety of types of links, ranking criteria, and filters. As an example, in Figure 3 we show the graph view displaying the authors most similar to Enrico Motta according to the temporal topic similarity, a novel metric which reflects the similarity of people's research trajectories with respect to the temporal sequencing of their research interests. The radius of the nodes in the graph reflects the number of his/her publications in the Semantic Web area. Other author ranking measures, as discussed in the previous section, can also be used. Users can also choose from six types of relations between authors: co-publication, co-citation, topic similarity, temporal topic similarity, publication pattern similarity and citation pattern similarity.

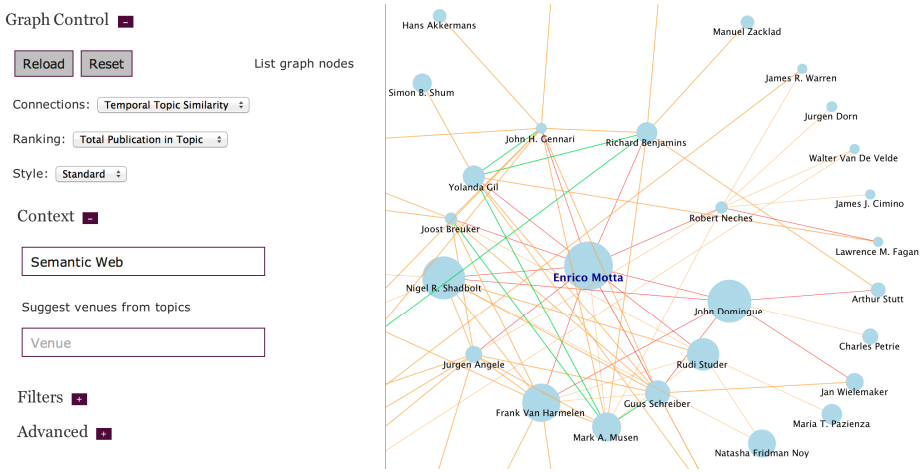


Fig. 3. A graph view in Rexplore

The topic similarity reflects how similar two authors are with respect to their research topics and takes advantage of the fine-grained topic structure generated by Klink and its semantic characterization. A naïve way to compute it would be to directly compare the vectors representing the number of publications associated with a keyword. However, treating topics as strings, as many systems do, would yield poor result. In fact, keywords referring to a related area, to a sub area, or even indicating the same topic with a different name would be considered different. For example, a prominent author in the field of Linked Data would have most of his or her publications associated to a "Linked Data" keyword, and may be considered uncorrelated to authors who have papers tagged as "Semantic Web". Thus we exploit a variation of the semantic enrichment procedure already mentioned in Section 3.3 on the publication vectors of the authors, assigning each publication on a topic also to its *broaderGeneric* or *relatedEquivalent* topics. However, in this case we want also to include *contributesTo* relationships, which yield important information but cannot be handled in the same way. In fact, it is not automatic that a paper published under a

topic (e.g., "Ontology Engineering") is also about its *contributesTo* topics (e.g., "Semantic Web"). It seems however appropriate to use the probability that the contributing topic T_1 refers to a certain topic T_2 to assign an additional bonus to T_2. Thus, in case of a *contributesTo*(T_1,T_2) relationship, we assign to T_2 only a fraction of the publications in T_1 according to the formula:

$$CT(T) = \sum_{i=1}^{n} P\big(T \big| ct(i,T)\big)^{\alpha}$$

where T is a topic, n is the number of publications of an author that are not already associated with T but have at least one topic in a *contributesTo* relationship with T, $ct(i,T)$ is the set of topics associated with the *i-th* publication that are in a *contributesTo* relationship with T, $P\big(T \big| ct(i,T)\big)$ is the probability for a paper with the set of topics $ct(i,T)$ to be also explicitly associated with area T (or with a topic having a *broaderGeneric* or *relatedEquivalent* relationship with T) before the publication date of the *i-th* paper and α is a factor which modulates the *contributesTo* relationship (empirically set to 0.5 in the prototype).

By taking into account the publication date of each paper, the formula considers also the changes in topic relations over time. For example a paper about "Ontology Engineering" in the year 2001 would have a lower probability to be about "Semantic Web" than a paper about the same topic in 2010 and thus should contribute much less to "Semantic Web" in the author publication vector. The topic similarity is finally computed as the cosine similarity between the semantically enriched vectors of publications.

The temporal topic similarity builds on the topic similarity measure and makes it possible to identify groups of researchers who appear to be following similar research trajectories, sharing research interests and moving from one topic to another in a similar way. In particular, this is very useful to identify the various sub-communities that populate a particular research area. The temporal topic similarity takes into account the order and the time span in which an author has published on a certain topic and is calculated as the weighted average of the topic similarities computed on different time intervals. Thus, if author A worked on T_1 and then moved to T_2, he or she may be similar to author B who was originally in T_2 and then moved to T_1 in terms of topic similarity, but will be different in terms of temporal topic similarity.

Finally, the publication/citation pattern similarity reflects how similar two authors are with respect to their career progression in terms of number of publications/ citations.

The graph view also provides a variety of standard interface operations, such as changing the level of granularity in the view, expanding, closing, or hiding nodes, etc. In addition, both nodes and links can be filtered with respect to specific years, topics, and venues. For example, it is possible to customize a graph and visualize only the co-authors of a particular researcher, who have between 5 and 15 career years, have published in both Linked Data and Social Networks, and have publications in CHI.

4 Empirical Evaluation

4.1 Experimental Setup

For the evaluation we enrolled 17 PhD students and researchers drawn from the members of the Knowledge Media Institute in UK and the University of Turin in Italy. None of these subjects had been involved in the work on Rexplore, or indeed knew the system prior to the evaluation session. At the beginning of the evaluation session, every subject filled a questionnaire about his/her research experience, topics of interest, and familiarity with a list of systems that included Google Scholar (GS), MAS, DBLP, and Citeseerx. This was followed by a 15-minutes tutorial about Rexplore and then the subjects were asked to perform the activities listed in Table 1.

Table 1. Activities in the Evaluation Process

Activity 1. Carry out the tasks shown in Table 2 using Rexplore.
Activity 2. Select one of the three tasks in Table 2 and attempt to achieve it using either Google Scholar (GS) or Microsoft Academic Search (MAS).
Activity 3. Suggest a task you would consider valuable and perform it using Rexplore.

Table 2. Evaluation Tasks

Warm-up Task. Find the 3 main co-authors (in any field) of the author with most publications in the topic User Modelling.
Task 1. Find the top 3 'rising stars' in the United Kingdom with expertise in both *Semantic Web* and *Social Networks*, in the career range 5-15 years from first publication, ranked in terms of number of citations in these 2 areas.
Task 2. Find the top 5 authors with the highest number of publications in the *Semantic Web* and rank them in terms of number of publications in *Artificial Intelligence*. For each of them find their most cited paper in *Artificial Intelligence*.
Task 3. Which are the 2 sub-topics in *Semantic Web* that have grown the most in 2005-2010 (as measured by the difference between the number of papers in 2010 and in 2005) and who are the top 2 authors (ranked by number of publications in topic) in these 2 topics.

The rationale for selecting GS and MAS as control systems was that GS is the most widely used bibliographic search engine, while MAS provides a number of features, in terms of time-based visualizations, which go well beyond what is provided by GS.

Each task was recorded with screen capturing software and the time taken for completion was measured; if a task was not solved within 15 minutes, it was recorded as 'failed'. Tasks not completed within the time limit were considered as 15 minutes performance. After completing the various tasks, the 17 participants were requested to fill in a usability SUS questionnaire [16] and a second questionnaire about the strengths/weaknesses of the tested systems. On average the total time required to complete each evaluation session was slightly less than 2 hours.

In contrast with other evaluation studies –e.g., see [17], where participants were divided in different groups and each group would use a different tool to perform the same set of tasks, here we did not carry out a straightforward 'tool shootout', but we instead implemented a more faceted experimental design, comprising usability questionnaires and a task-centric evaluation, and also providing the participants with the opportunity both to suggest their own tasks and also to try out other tools. The reason for this is that GS and MAS do not directly support the kinds of sensemaking tasks for which Rexplore offers support, hence a 'tool shootout' would have provided little valuable data and most likely caused a high degree of frustration for the subjects. For this reason we decided to focus the bulk of the evaluation on identifying opportunities to evaluate and gather feedback on Rexplore, while still collecting some comparative data.

The tasks given to the subjects cover common scenarios to do with expert search and trend detection. Task 1 is a common expert search task –e.g., for research leaders who wish to identify 'new blood' to fill a certain position. Task 2 is also a common expert search task, where, given a pool of people with expertise in topic A, we want to identify the person in the pool that can be considered as the top expert in topic B. Task 3 is about detecting trends and analyzing research topics. It is a common task for many professionals, such as managers in research funding bodies, who may wish to identify which areas appear to be particularly 'hot' within a broader research field.

4.2 Results

In Activity 1, the 17 subjects were able to complete within the requested 15 minutes 50 of the 51 (17*3) tasks using Rexplore, with a 98% success rate. The only failure was registered in Task 2. Task 1 was the simplest one and was performed on average in about 3 minutes. In fact this task required only the ability to formulate a complex query, followed by the manual identification of a number of authors. Task 2 required a more complex exploration of the system, since the user had to first select five authors and then explore them using the graph view or the author analysis page, to find out their contributions in Artificial Intelligence. Task 3 required the use of visual charts showing the publication trends of the sub-topics and the use of the topic navigator to identify the best authors.

Table 3. Experimental results (in min:secs) using Rexplore and MAS. The tasks performed with GS yielded no success, thus their average time is by definition equal to 15:00.

	Rexplore (N=17)			**MAS (N=9)**		
	Task 1	**Task 2**	**Task 3**	**Task 1 (N=6)**	**Task 2 (N=2)**	**Task 3 (N=1)**
Average Time	3:06	8:01	7:51	14:46	13:52	15:00
Standard Dev.	0:45	2:50	2:32	0:24	1:35	00:00
Success Rate	100%	94%	100%	33%	50%	0%

In Activity 2, eight subjects were asked to work with GS and nine with MAS. Task 1 was chosen by 6 users on MAS and 5 on GS, while Task 2 was chosen respectively by 2 and 3 subjects. Task 3 was perceived by the subjects as practically impossible to do with a system without a fine-grained topic analysis functionality, and as a result was tried only by one subject (using MAS). Only three people out of nine completed a task with MAS (overall 33% success rate) and none at all with Google Scholar. Hence, the success rates of the three systems are significantly different: the two by two table comparison between Rexplore and MAS analyzed with a Fisher test (a standard statistical significance test used in the analysis of contingency tables when the numbers involved are small) yields $p=10^{-5}$, whereas the three by two table including also GS yields $p<10^{-7}$. Incidentally, the users who were able to complete the chosen task on MAS (2/8 for Task 1 and 1/2 for Task 2) were among the best performers in Activity 1, and required for the same task about 5 times longer on MAS than on Rexplore, even after having already successfully completed the task in Rexplore. Table 3 summarizes the time employed for the assigned tasks on Rexplore and MAS and the relative success rate –i.e., the number of jobs completed correctly within 15 minutes.

An important question when using a tool for navigating a research area is how much prior knowledge of the domain affects task performance. The results of the evaluation show that the average time for completing the three tasks by subjects with expertise in Semantic Web (that is the main area of the tasks) is not significantly different from the one obtained by the others ($p=0.63$ according to the t-test). However, the experts in tools for exploring academic data, who are active in fields such as Bibliometric and Learning Analytics, were instead able to get acquainted with the Rexplore system much more quickly and use it more effectively than the other subjects. The average time of the former group on the three tests was 5:01±0:02 min, against 6:52 ± 0:06 min of the latter ($p < 0.022$). On the contrary, no correlation was found with the usage of other tools for academic exploration, such as GS, MAS, DBLP, ACM, Citeseerx and Scopus. Hence the data appears to show that no domain-specific expertise is needed to use Rexplore to make sense of a particular research area, while at the same time the tool does not penalize experts in Bibliometrics and Learning Analytics, who are used to carrying out these kinds of analyses.

The tasks proposed by the subjects in Activity 3 were a good mix of routine searches and creative queries, and thus the performances cannot be directly compared. 59% of the subjects chose to investigate a single author, using mainly the topic analysis and the graph view, whereas 23% of them preferred to explore a research area to understand better its migration patterns and trends. The ability to filter by multiple topics or using them for author analysis was widely appreciated: 71% of the proposed tasks involved topic filtering or topic analysis on an author or group of authors. The integration of the different Rexplore functions made it possible to try particularly interesting exploration tasks: for example, a particularly creative subject tried to find a better affiliation for an author by analyzing organizations, topic similarity and prominence of the researchers connected to him through the various links provided by Rexplore (incidentally, he opted for MIT).

15 out of the 17 subjects considered their suggested task satisfactorily concluded. One of them was unable to complete her proposed task because of problems with the original data tagging: the subject was searching for papers of a certain author about Semantic Web, which were actually tagged only as "Knowledge Base" in the original data which Rexplore uses. This suggests that relying exclusively on user-defined keywords may not be sufficient and even when these are available, it may be useful to refine them by analyzing the abstract or the full text of the paper.

Rexplore reached a score of 75/100 on the standard SUS usability test, based on ten multiple-choice questions. A score of 75 can be converted to a percentile rank of 72%, meaning that the usability of Rexplore was considered equal or superior than 72% of the 500 tested systems. In particular 94% of the subjects agreed or strongly agreed on the fact that the functions of the system are well integrated and 82% stated that they would be happy to use Rexplore for their work.

The post-task questionnaire included three sections. In the first and second parts the users were asked for their opinions about the support given by Rexplore and GS/MAS for the assigned tasks. In the third part they were asked to comment about the support provided by Rexplore for the task suggested by them.

In the first section, 94% of the subjects described Rexplore as "very effective", while 18% described it as "easy/natural/intuitive". Among the most useful features were the faceted filters (59%), the visualization/charts (47%), the graph view (47%) and the semantic characterization of topics (41%). The main weaknesses of Rexplore were found to be its visual complexity (41%) and a not always well-evidenced navigation context (35%). Indeed, according to some users, the high number of functionalities offered by Rexplore may also be overwhelming.

When asked to suggest new features that would facilitate their exploration of academic data, 23% of subjects suggested some "minor interface change", especially in the direction of solving the aforementioned problem of "making the context clearer in any moment" (18%). 23% of them thought Rexplore did not need any additional features and 18% proposed additional filters. Other features that the users suggested include a natural language interface for formulating complex searches and the ability to retrieve and search the full text of a publication from within Rexplore.

Trying to perform the kind of task described in Table 2 with MAS or GS frustrated the users: 88% of the subjects using MAS and 89% using GS described the support of those systems as "ineffective". The reasons of their frustration were various: not effective contextual filtering (77% MAS, 65%GS), absence of semantic/structured topics (56% MAS, 63%GS), and poor support for complex/multidimensional queries (33% MAS, 50% GS). Finally, the support provided by Rexplore for user-defined tasks (Activity 3) was also rated positively. 76% of the participants defined such support as "effective/very effective/unique", while 12% of them, though they were able to complete their task, found some "minor problems", usually to do with missing filter options. Indeed it seems that users could do with a variety of filters well beyond what it is normally considered in these systems: one of the subjects suggested a filter able to discriminate genders, while another asked to be able to split publications by the particular author position (e.g., first or second author). Nonetheless the results of the evaluation appear very satisfactory, confirming that Rexplore provide a degree of support that users consider effective and valuable for performing real-world tasks.

5 Conclusions

In this paper we have presented Rexplore, a novel tool for exploring scholarly data, which integrates a semantic foundation with statistical and visual analytics solutions to support users in exploring and making sense of scholarly data. The results from the empirical evaluation confirm the effectiveness of the functionalities provided by the tool and show a high value of user satisfaction. In particular, users rate very highly the semantic underpinning of the tool, which arguably affords a major advantage over other tools in its ability to support i) the visualization of trends at a very fine level of granularity, ii) methods to identify 'semantic' relations between authors, and iii) fine-grained multi-dimensional academic expert search.

For the future we plan to extend the tool by enhancing its functionalities through the integration of other sources of data relevant to academic activities and we also aim to address the minor interface issues identified during the evaluation. We also plan to add to the number of navigation filters, a feature which users appear to value extremely high. Finally, we are actively discussing with a number of commercial providers of scholarly data, with the aim to release a version of the tool with comprehensive data coverage for use by the scientific community.

References

1. Dunne, C., Shneiderman, B., Gove, R., Klavans, J., Dorr, B.: Rapid understanding of scientific paper collections: Integrating statistics, text analytics, and visualization. American Society for Inf. Science and Technology 63(12), 2351–2369 (2012)
2. Motta, E., Osborne, F.: Making Sense of Research with Rexplore. In: 11th Int. Semantic Web Conference, Poster&Demo Session, Boston, MA (2012)
3. Diederich, J., Balke, W.T., Thaden, U.: Demonstrating the Semantic Growbag: Automatically Creating Topic Facets for FacetedDBLP. In: Proceeding of the 7th ACM/IEEE-CS Joint Conference on Digital Libraries (2007)
4. Li, H., Councill, I., Lee, W.C., Giles, C.L.: CiteSeerx: an architecture and web service design for an academic document search engine. In: Proceedings of the 15th Int. Conference on the World Wide Web, pp. 883–884 (2006)
5. Tang, J., Zhang, J., Yao, L., Li, J., Zhang, L., Su, Z.: ArnetMiner: extraction and mining of academic social networks. In: Proceeding of the 14th Int. Conference on Knowledge Discovery and Data Mining, pp. 990–998 (2008)
6. Monaghan, F., Bordea, G., Samp, K., Buitelaar, P.: Exploring Your Research: Sprinkling some Saffron on Semantic Web Dog Food. In: Semantic Web Challenge at the International Semantic Web Conference (2010)
7. Möller, K., Heath, T., Handschuh, S., Domingue, J.: Recipes for semantic web dog food—The ESWC and ISWC metadata projects. In: Aberer, K., Choi, K.-S., Noy, N., Allemang, D., Lee, K.-I., Nixon, L.J.B., Golbeck, J., Mika, P., Maynard, D., Mizoguchi, R., Schreiber, G., Cudré-Mauroux, P. (eds.) ASWC 2007 and ISWC 2007. LNCS, vol. 4825, pp. 802–815. Springer, Heidelberg (2007)
8. Yee, P., Swearingen, K., Li, K., Hearst, M.: Faceted Metadata for Image Search and Browsing. In: Proceedings of CHI 2003, pp. 401–408. ACM (2003)

9. Hildebrand, M., van Ossenbruggen, J., Hardman, L.: /facet: A browser for heterogeneous semantic web repositories. In: Cruz, I., Decker, S., Allemang, D., Preist, C., Schwabe, D., Mika, P., Uschold, M., Aroyo, L.M. (eds.) ISWC 2006. LNCS, vol. 4273, pp. 272–285. Springer, Heidelberg (2006)
10. Schraefel, M.C., Wilson, M., Russell, A., Smith, D.A.: mSpace: improving information access to multimedia domains with multimodal exploratory search. Communications of the ACM 49(4), 47–49 (2006)
11. Popov, I.O., Schraefel, M.C., Hall, W., Shadbolt, N.: Connecting the dots: a multi-pivot approach to data exploration. In: Aroyo, L., Welty, C., Alani, H., Taylor, J., Bernstein, A., Kagal, L., Noy, N., Blomqvist, E. (eds.) ISWC 2011, Part I. LNCS, vol. 7031, pp. 553–568. Springer, Heidelberg (2011)
12. Bergstrom, P., Atkinson, D.C.: Augmenting the exploration of digital libraries with web-based visualizations. In: ICDIM 2009. IEEE (2009)
13. Osborne, F., Motta, E.: Mining semantic relations between research areas. In: Cudré-Mauroux, P., Heflin, J., Sirin, E., Tudorache, T., Euzenat, J., Hauswirth, M., Parreira, J.X., Hendler, J., Schreiber, G., Bernstein, A., Blomqvist, E. (eds.) ISWC 2012, Part I. LNCS, vol. 7649, pp. 410–426. Springer, Heidelberg (2012)
14. Bizer, C., Lehmann, J., Kobilarov, G., Auer, S., et al.: DBpedia-A crystallization point for the Web of Data. Journal of Web Semantics 7(3), 154–165 (2009)
15. Osborne, F., Motta, E.: Exploring Research Trends with Rexplore. D-Lib Magazine 19(9/10) (2013)
16. Brooke, J.: SUS: A “quick and dirty” usability scale. In: Jordan, P.W., et al. (eds.) Usability Evaluation in Industry, pp. 189–194. Taylor & Francis, London (1996)
17. Motta, E., Peroni, S., Gómez-Pérez, J.M., d’Aquin, M., Li, N.: Visualizing and Navigating Ontologies with KC-Viz. In: Proceedings of the 10th Int. Semantic Web Conference, pp. 343–362. Springer (2011)

Personalized Best Answer Computation in Graph Databases

Michael Ovelgönne[1], Noseong Park[2], V.S. Subrahmanian[1,2], Elizabeth K. Bowman[3], and Kirk A. Ogaard[3]

[1] UMIACS, University of Maryland, College Park, MD, USA
[2] Department of Computer Science, University of Maryland, College Park, MD, USA
[3] Tactical Information Fusion Branch, Computational and Information Sciences, U.S. Army Research Laboratory, Aberdeen Proving Ground, MD, USA
{mov,npark,vs}@umiacs.umd.edu,
{elizabeth.k.bowman.civ,kirk.a.ogaard.ctr}@mail.mil

Abstract. Though subgraph matching has been extensively studied as a query paradigm in semantic web and social network data environments, a user can get a large number of answers in response to a query. Just like Google does, these answers can be shown to the user in accordance with an importance ranking. In this paper, we present scalable algorithms to find the top-K answers to a practically important subset of SPARQL-queries, denoted as *importance queries*, via a suite of pruning techniques. We test our algorithms on multiple real-world graph data sets, showing that our algorithms are efficient even on networks with up to 6M vertices and 15M edges and far more efficient than popular triple stores.

1 Introduction

Facebook recently introduced a new feature called "graph search"[1] that enables users to search Facebook's social graph. This graph contains entities like persons, media items, companies, events, and associated data of these entities like name or age. For example, users can search for *cities that friends of their parents like* or *restaurants their friends have been to*. Such queries are a special case of SPARQL queries and of the class of subgraph matching queries (for the first example the pattern is the path graph *user* $\leftrightarrow$ *parent* $\leftrightarrow$ *friend* $\leftrightarrow$ *city*) with additional constraints on the vertex properties.

In this paper, we go beyond subgraph matching and consider the case of subgraph queries augmented with "importance" metrics that are specified by the user in his query. Such queries can be easily expressed in SPARQL using FILTER and ORDER BY clauses. In classical subgraph queries, the user specifies a query subgraph – and all matches of that subgraph with subgraphs of the graph database are considered equally important. However, when the nodes in the graph have associated semantic labels, then there are cases where the user may specify an importance measure that marks some matches as being "more important" than others.

[1] https://www.facebook.com/about/graphsearch

H. Alani et al. (Eds.): ISWC 2013, Part I, LNCS 8218, pp. 478–493, 2013.

A query for restaurants a person's friends have been to can return hundreds or thousands of answers, as many Facebook users have hundreds of friends. However, users will prefer a short list of the *most relevant* restaurants. We want to provide users a tool to find the most relevant answers from their perspective. An *importance query* is an extended subgraph query with a scoring mechanism, and as such they are a subset of SPARQL queries. With importance queries users can, e.g. search for restaurants with the *highest star ranking* their friends like or the *largest* cities friends of the parents have been to.

Figure 1 shows a Facebook-style graph with four types of edges (friend of, resident of, located in, likes). Each vertex in this graph has different kinds of properties such as the type of the vertex (person, restaurant, city), the age and gender of persons, and the star rating of restaurants. A possible query on this graph is: Which are the restaurants with the highest star ranking in London that my friends who live in London like?

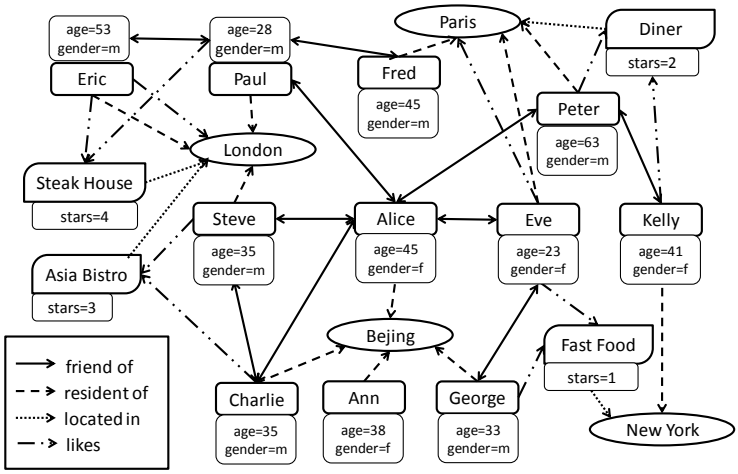


Fig. 1. Example of a data graph

Our technical solution for finding the most important pattern matches in a graph is based on classic subgraph matching algorithms rather than join-based techniques for RDF database engines. Especially because of expensive join operations, finding patterns in large graph-structured datasets stored as triples is inefficient [16]. Our experimental evaluation compares the newly developed algorithms to RDF triples stores, and shows that our algorithms beat them on importance queries.

An important point to note is that in this paper we only consider *anchored* queries, i.e. subgraph queries where we already know the mapping for at least one of the vertices in the query. This is often a more realistic problem setting compared to arbitrary, non-anchored queries, because people usually create

searches with themselves as an anchor (queries containing terms like *my* friends, *my* company, cities *I* like).

In this paper we extend the subgraph matching problem and try to find the *most important* matches (according to a user provided definition) in attributed graphs (i.e. graphs with edge labels and where vertices may have associated properties). We make the following contributions.

- First, we formally define importance queries and define answers (and the top-k answers) to such queries (Section 2).
- We then define a simple baseline algorithm to solve such queries, followed by our more sophisticated OptIQ algorithm that can efficiently prune part of the search space and scale our top-k algorithms to find answers to importance queries (Sections 3 and 4).
- We present the results of experiments to analyze the influence of query properties on the performance of query algorithms (Section 5). Our experiments – on CiteSeerX, YouTube, Flickr and GovTrack data, show that our algorithms scale well to data sets containing up to 6.2M vertices and 15.2M edges. We also show that popular triple stores are much slower in answering importance queries.

2 Importance Queries on Graphs

In this section, we formalize the concept of *importance queries* – the query type we developed fast answering algorithms for. We use the following notation in this paper. $\mathbb{R}$ denotes the set of all non-negative real numbers. VP, EP, and VAR are arbitrary but fixed mutually disjoint sets of symbols for *vertex predicates*, *edge labels* and *variables*, respectively. Variable symbols start with a "?" (e.g. ?x). Every vertex predicate $p \in \mathsf{VP}$ has a *domain* $dom(p)$ which is some set disjoint from each of $\mathsf{VP}, \mathsf{EP}, \mathsf{VAR}$.

Definition 1 (Graph Database). *A* graph database *(GDB) is a triple* $\mathcal{G} = (V, E, \wp)$ *with* V *a finite set of* vertices*,* $E \subseteq V \times \mathsf{EP} \times V$ *a finite set of labeled edges and* $\wp : V \times \mathsf{VP} \to \bigcup_{p \in \mathsf{VP}} dom(p)$ *a* property function*. We assume that for all* $v \in V, p \in \mathsf{VP}$*,* $\wp(v, p) \in dom(p)$*.*

$V_\mathcal{G}, E_\mathcal{G}, \wp_\mathcal{G}$ denote the vertices, edges, and property functions of a GDB $\mathcal{G}$. Throughout this paper, we assume that $\mathcal{G} = (V, E, \wp)$ is an arbitrary but fixed graph. Figure 1 shows a sample of such a GDB.

Definition 2 (Term; Numeric Term). *(i) Every member of* $\bigcup_{p \in \mathsf{VP}} dom(p)$ *is a term. If* $nt \in \bigcup_{p \in \mathsf{VP}} dom(p) \cap \mathbb{R}$*, then* nt *is a* numeric term*.*
(ii) If $?x \in \mathsf{VAR}$ *and* $p \in \mathsf{VP}$*, then* $?x.p$ *is a term. If* $dom(p) \subseteq \mathbb{R}$*, then* $?x.p$ *is a* numeric term*.*
(iii) If nt_1, nt_2 *are numeric terms, then* $nt_1 + nt_2$ *and* $nt_1 * nt_2$ *are numeric terms.*
A term is ground *if no variables occur in it. We say a term* t *is* solely about variable $?x$ *if* $?x$ *is the only variable occurring in* t*.*

The importance query (definition follows) shown in Figure 2 contains the numeric term $?r.stars$. We assume that all *ground* numeric terms are evaluated, e.g. the numeric term $2 + 3$ is evaluated to 5.

Definition 3 (Constraint). *(i) If t_1, t_2 are terms, then $t_1 = t_2$ and $t_1 \neq t_2$ are constraints.*
(ii) If nt_1, nt_2 are numeric terms, then $nt_1 < nt_2, nt_1 \leq nt_2, nt_1 > nt_2, nt_1 \geq nt_2$ are constraints.
(iii) If c_1, c_2 are constraints, then $c_1 \wedge c_2$ is a constraint.
We say constraint C is solely about variable $?x$ *if $?x$ is the only variable occurring in C.*

The example importance query in Figure 2 contains for variable $?r$ the constraint $?r.type = restaurant$.

Definition 4 (Importance Query). *An* importance query *is a 4-tuple $PQ = (SQ, \chi, \varrho, agg)$ where:*

1. *SQ is a pair $SQ = (QV, QE)$ where $QV \subseteq V \cup \mathsf{VAR}$ and $QE \subseteq (V \cup \mathsf{VAR}, \mathsf{EP}, V \cup \mathsf{VAR})$. Because V and VAR are finite sets, QV and QE are finite sets as well. SQ is called a* subgraph query.
2. *χ associates a constraint that is solely about $?x$ with each variable $?x \in QV \cap \mathsf{VAR}$.*[2]
3. *ϱ is a partial function from $QV \cap VAR$ to numeric terms s.t. there is at least one $?x \in QV \cap VAR$ which is mapped to a numeric term with $?x$ occurring in it.*
4. *agg is one of four aggregation function MIN, MAX, SUM or AVG.*[3]

Suppose $SQ = (QV, QE)$ is a subgraph query. A *substitution* is a mapping $\theta : QV \cap \mathsf{VAR} \to V$. Thus, substitutions assign vertices in a GDB $\mathcal{G}$ to variables in QV. The *application* of a substitution θ to a term t, denoted $t\theta$, is the result of replacing all variables $?x$ in t by $\theta(?x)$. When t contains no variables, then $t\theta = t$.

If we consider the sample query Q shown in Figure 2, it has two answers w.r.t. the graph database shown in Figure 1:

$$\theta_1 \equiv ?p = Steve, ?r = AsiaBistro$$
$$\theta_2 \equiv ?p = Paul, ?r = SteakHouse$$

Definition 5 (Answer; Answer Value). *Suppose $\mathcal{G}$ is a GDB, $PQ = (SQ, \chi, \varrho, agg)$ is an importance query, and θ is a substitution w.r.t. SQ. θ is an* answer *of PQ w.r.t. $\mathcal{G}$ if:*

[2] If we do not wish to associate a constraint with a particular variable $?x$, then $\chi(?x)$ can simply be set to a tautologous constraint like $2 = 2$.
[3] These functions map multisets of reals to the reals and are defined in the usual way.

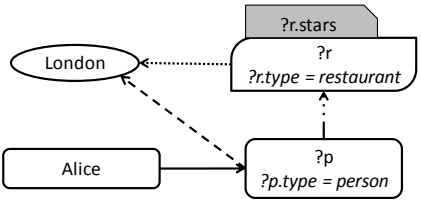


Fig. 2. Example of an importance query described by a subgraph query, constraints (italic) and an IQ-term (gray box)

(i) for every edge $(v_1, ep, v_2) \in QE$*, it is the case that* $(v_1\theta, ep, v_2\theta) \in E$ *and (ii) for each vertex* $?x \in QV \cap \mathsf{VAR}$*, the constraint* $\chi(?x)\theta$ *is true.*

The answer value *of a substitution* θ*, denoted* $\mathsf{Aval}(\theta, PQ, \mathcal{G}) = agg(\{(\varrho(?x))\theta \mid ?x \in dom(\varrho)\})$*. When the set on the right hand side is empty,* $\mathsf{Aval}(\theta, PQ, \mathcal{G}) = 0$*.*

We use $\mathsf{ANS}(PQ, \mathcal{G})$ *to denote the set of all answers of importance query* PQ *w.r.t. GDB* $\mathcal{G}$*.*

In our example ϱ assigns the very simple IQ-term $?r.stars$ to the variable $?r$. So the answer value of θ_1 is 3 and the answer value of θ_2 is 4.

3 Baseline Best Answer Algorithm

In Section 2 we defined importance queries. Depending on the size of the data graph and the constrained IQ-query, the number of results can be very large. We defined the notion of importance queries as users are usually only interested in the most important query answers. Consequently, we will discuss top-k query answer algorithms.

A straightforward algorithm to compute the answers of an importance query follows the definition of importance queries and first computes all subgraph query answers, filters the set of answers to those answers that satisfy the constraints and then computes the IQ-values. Any subgraph matching algorithm could be used here (see Sec. 6). However, we use an implementation that considers our specific problem situation (queries with anchors and large disk-residing graphs). Subgraph matching algorithms are branch-and-bound algorithms that follow a search tree. In our case, first, an anchor is selected. Then an unmapped neighbor of an anchor or a mapped variable in the query graph gets selected, and all candidates for this variable in the data graph are determined. For every candidate the variable is mapped to the candidate, and the search with the next unmapped variable is continued recursively. We only use the I/O- efficient pruning on vertex degrees because determining vertex degrees does not require one to read extra data. Loading index data for advanced indexes from disk usually does not pay off.

Algorithm 1. Optimized Importance Query (OptIQ) Algorithm

1 FUNCTION `AnswerQuery`
Input: Data Graph G, Importance Query $q = (SQ = (QV, QE), \chi, \varrho, agg)$, partial substitution θ, result size k
Output: answered stored in global variable A: set of tuples $(vertex, score)$
2 **if** θ *maps every variable to a ground term* **then**
3 $A \leftarrow A \cup \{\theta\}$
4 **if** $|A| > k$ **then**
5 $A \leftarrow A \setminus \{\theta \in A$ with minimal `score(`θ`)` $\}$
6 $nextvars \leftarrow \{(c, ?v) | ?v \rightarrow c$ or $c \rightarrow ?v \in QE\}$ //edges with one mapped endpoint
7 **foreach** $(c, ?v) \in nextvars$ **do**
8 $R_{c,?v} \leftarrow$ `getNeighborNum(`G, c, `getEdgeLabel(`$(c, ?v)$`))`
9 $B_{c,?v} \leftarrow$ `getExpBenefit(`G, q, $(c, ?v), \theta$`)` // for WCOST only
10 **if** $R_{c,?v} = 0$ **then return**
11 $(c, ?w) \leftarrow (c, ?v) \in nextvars$ with max $B_{c,?v}$ // for WCOST
12 with min $R_{c,?v}$ // otherwise
13 $N_{?w} \leftarrow$`GetValidNeighbors(`G, q, $(c, ?w)$`)`
14 **foreach** $m \in N_{?w}$ *in decreasing order of score* $\varrho(m)$ **do**
15 $\theta' \leftarrow \theta \cup (?w \rightarrow m)$
16 $s \leftarrow$ `calculateMaxScore` (G, θ')
17 **if** $|A| > k$ *and* $s <$ *lowest score of any* $\theta \in A$ **then** continue
18 `AnswerQuery(`G, $q\theta'$,θ',k`)`

19 FUNCTION `GetValidNeighbors`
Input: Data Graph G, query q, tuple (vertex c, variable $?w$)
Output: vertices that can be mapped to $?w$ among all c's neighbors

20 FUNCTION `getNeighborNum`
Input: Data Graph G, vertex c, edge label l
Output: The number of c's neighbors which are connected through an edge of the label l

21 FUNCTION `getExpBenefit`
Input: Data Graph G, query q, tuple (vertex c, variable $?w$), partial mapping θ
Output: `getExpScore` $(G, q, (c, ?w), \theta)$ / `getCost` $(G, (c, ?w))$

22 FUNCTION `getExpScore`
Input: Data Graph G, query q, tuple (vertex c, variable $?w$), partial mapping θ
Output: $agg(\{?v \in QV \cap VAR : value(?v)\})$, where $value(?v) = \varrho(?v)\theta$ if θ maps $?v$, $value(?v) =$ `localAvg`$(?v)$ if all candidates for $?v$ are in a known, cached subgraph of the subgraph index of G, and $value(?v) = 0$ otherwise

23 FUNCTION `getCost`
Input: Data Graph G, tuple (vertex c, variable $?w$)
Output: $n \log n$, where n=`getNeighborNum` $(G, (c, ?w))$, i.e. sorting time in l. 14

24 FUNCTION `calculateMaxScore`
Input: Data Graph G, partial mapping θ
Output: $agg(\{?v \in QV \cap VAR : value(?v)\})$, where $value(?v) = \varrho(?v)\theta$ if θ maps $?v$, $value(?v) =$ `localMax`$(?v)$ if all candidates for $?v$ are in a known, cached subgraph of the subgraph index of G, and $value(?v) =$ `globalMax`$(?v)$ otherwise

4 Optimized (OptIQ) Algorithm

The baseline algorithm (Sec. 3) performs the 4 steps (1) Subgraph Matching, (2) Constraint Checking, (3) Scoring and (4) Top-k Selection sequentially and independently. An obvious improvement is the integration of the uncoupled steps. If we check the constraints in the subgraph matching step, then we do not have to create a possibly large list of subgraph matches that needs to be checked for meeting the constraints. Likewise, we can maintain a sorted list of top-k substitutions. Every time a new substitution with a score greater than the lowest score in the top-k list has been identified, we update the list.

Algorithm 1 shows the integrated algorithm. All blue code segments are extensions to improve the performance, but are not necessary to compute answers to IQ-queries *per se*. We will discuss these improvements in Sections 4.2–4.4.

Lines 2–5 check whether a complete substitution has been generated, and add a complete substitution to the answer set if its score is among the top-k. Lines 6–10 inspect every edge of the query graph whose one end is mapped to a vertex of the data graph and whose other end is not. In $R_{c,?v}$, we store the number of c's neighbors in the data graph that are connected through an edge with the same label between c and $?v$, i.e. $R_{c,?v}$ is the number of candidates for $?v$. In line 11 we select the query graph edge with the lowest number of candidates. `GetValidNeighbors`() returns the set of all valid vertices that can be mapped to $?w$. Here, we use DOGMA's pruning technique based on IPD values (see Section 4.1) to filter neighbors that cannot be part of a valid answer. Other pruning strategies (see e.g. [4, 13, 14]) could be used as well. In line 14–18, we substitute $?w$ with each candidate m and recursively continue the assembly of answers.

Before we can discuss the performance improving techniques shown in the blue code segments of Algorithm 1, we need to introduce our graph database index.

4.1 Database Index

To efficiently answer importance queries on large graphs, we use a disk-based index inspired by the DOGMA index [3]. We decompose the data graph into a large number of small, densely connected subgraphs and store them in an index. Our partitioning algorithm follows the multi-level graph partitioning scheme [7]. Like DOGMA, we iteratively halve the number of vertices by merging randomly selected vertices with all of their neighbors. When the resulting graph has less than 100 vertices, we iteratively expand the graph using the GGGP algorithm from the METIS algorithm package [5] to bisect the graph components at each level.[4] For every block of the partition we extract the subgraph it induces from the graph database and store it as one block of data to disk.

The objective of the DOGMA index is two-fold: (1) to increase the I/O-efficiency by exploiting data locality – only those parts of the graph that are

[4] We also conducted preliminary experiments with other partitioning algorithms but they showed no significant difference for the query processing performance.

necessary to answer a query have to be retrieved from disk (2) DOGMA stores for every vertex the internal partition distance (IPD), i.e. the number of hops from a vertex to the nearest other vertex outside the subgraph. Using the IPD, we can quickly compute a minimum distance between vertices, and prune candidates if their distance is higher than the distance between their respective query graph variables.

We extend this concept and store additional information in the index for advanced top-k pruning strategies. First, we store global maximum values for every vertex property. Additionally, we store together with each induced subgraph G_s the edges that connect it to other subgraphs (inter-subgraph edges) and aggregated information (maximum and average) of the predicate values of the vertices in G_s and of those vertices not in G_s but adjacent to a vertex in G_s (denoted as the *boundary* of G_s).

4.2 Simple Top-k Pruning on Scores

The optimized baseline algorithm does not exploit the fact that we are only interested in the top-k answers. During the stepwise assembly of substitutions, there will be partial substitutions which cannot make it into the top-k given the scores of the full substitutions that are already in the answer set. If we identify them, we can prune the respective branch of the search tree and save computation time.

First, the set $N_{?w}$ should be sorted by score in line 14. I.e., if $?w$ is scored by an IQ-term, $N_{?w}$ is sorted in decreasing order of the value of the term. This ensures that we evaluate the most promising candidates first.

The IQ-score of a substitution is the value of the aggregation function agg on the values assigned by the IQ-terms to the variables (see Def. 5). For a partial substitution θ, we can compute an upper bound of its answer value Aval by using upper bounds for $\varrho(?v)\theta$ of all unmapped variables. That means, we calculate an upper bound of the answer value by using the exact term score for every previously mapped variable and upper bounds for currently unmapped variables. This is performed by `calculateMaxScore`(). Our simple top-k pruning strategy uses precomputed global upper bounds, i.e. $max_{x \in V} \wp(x, p_i)$, for each vertex property p_i.

When the variance of vertex property values is high, using the global upper bound of a vertex property will not allow us to prune many branches of the search tree. A tighter upper bound is desirable. The mappings of the partial substitutions restrict the set of valid candidates for the currently unmapped variables. What we need is a fast way to find tight upper bounds for vertex property values given the mappings in the partial substitutions.

4.3 Advanced Top-k Pruning on Scores

In Section 4.2 we presented a simple top-k pruning strategy using upper bounds for the reachable substitution scores. Using the proposed database index, we can find tighter upper bounds that provide a higher pruning power.

For the candidate set $N_{?v}$ of $?v$, we can compute the upper bound for $\varrho(?v)$ using $max_{x \in N_{?v}} \wp(x, p_i)$. But computing the upper bound in this way would require us to read the property scores of all vertices in $N_{?v}$. This is prohibitively expensive because of the high costs of reading from disk. However, if we store the maximal property scores of a subgraph in the index, we can find a good upper bound in a reasonable amount of time.

In `calculateMaxScore()`, we compute the upper bound of the answer value of a partial mapping θ by computing the upper bound for each variable $?v$'s $\varrho(?v)$ (denoted as `value()`). If θ maps a variable $?v$ to a vertex of the data graph, we know the exact value of $\varrho(?v)\theta$. For a currently unmapped $?v$, we look at its distance to already mapped variables c, $dist(c, ?v)$. If $dist(c, ?v) < IPD(c)$ for some c, we know that $?v$ has to be mapped to the same subgraph as c. Then, we use `localMax` to compute `value` using the local maximum values of the subgraph of c. If $dist(c, ?v) < IPD(c) + 1$, we do the same but using the maximum values of the subgraph and its boundary. However, if $dist(c, ?v) > IPD(c) + 1$ for all c, we have no local information and `globalMax` computes `value` using global maximum values.

4.4 Processing Order

The baseline algorithm iteratively selects the unmapped variable with the smallest candidate set for processing. However, for importance queries this strategy sometimes leads to the late discovery of top-k answers. Selecting a variable with a higher number of candidates might not be bad when most candidates can be pruned very early. To weigh the different objectives (low number of branches to follow, following more promising paths first) we compute the benefit score $B_{c,?v}$ in line 9 and process candidates in decreasing order of their benefits. We define the benefit of substituting a variable $?w$ with n candidates in a partial substitution θ' as $wexp(\theta')/f(n)$, where $f(n)$ is the cost to process n candidates and $wexp(\theta')$ is the expected score of θ'. In Algorithm 1, `getExpBenefit()` calculates this score.

As in the case of computing upper bounds for substitution scores for pruning (Section 4.3), we compute $wexp(\theta')$ with the precomputed property scores of subgraphs. But additionally we weigh the expected term scores using the indegree of a candidate. The indegree is a simple heuristic for the probability that the variable will be mapped to a vertex. As I/O-efficiency is the primary problem of our algorithm, we use only information already read from disk to determine the expected score. Unavailable vertex property score estimates are replaced by 0.

To compute the expected value we proceed as follows. We classify unmapped variables in the query graph in two groups.

- If a variable $?v$ has no vertex c whose $hop(?v, c)$ is less than or equal to c's IPD value, we assume the property scores are 0 (or ∞ if the MIN aggregation is used). Computing an expected value would require reading many additional disk pages (which we want to avoid) or using global averages. But underestimating the real expected score is in this case favorable because it

puts variables whose mapping requires additional disk access at the end of the priority list.
- Otherwise, we use the weight expected value of the subgraph c is residing in. We know that all query variables whose distance from c is less than or equal to c's IPD value will be mapped in the same subgraph. So, we can use the precomputed weighted average property values of the subgraph as the expected value.

5 Experiments

In the following, we present an evaluation of the previously introduced top-k algorithms. We conducted experiments with 5 algorithm variants: the non-integrated baseline algorithm Base, the optimized importance query (OptIQ), the extension of OptIQ by simple top-k pruning (GMax), the extension of OptIQ by advanced top-k pruning (LMax), and the extension of LMax by the improved processing order (WCOST).

To see how our algorithms perform in relation to triple stores, we ran additional experiments using Apache Jena TDB 2.10.0 [1] and OWLIM-SE 5.3.5925 [8]. We considered using RDF-3x [10] as well, but had to omit RDF-3x because it cannot answer queries with cross-products because of a bug that still exists in the latest release. Importance queries can be easily written in SPARQL with its FILTER and ORDER BY clauses.

5.1 Experimental Setup

To evaluate our algorithms, we use four real-world datasets. Basic properties of these datasets are shown in Table 1.

Table 1. Evaluation datasets

Name	#Vertices	#Edges	#V.Prop.	#E.Labels
CiteSeerX	0.93M	2.9M	5	4
YouTube	4.6M	14.9M	8	3
Flickr	6.2M	15.2M	4	3
GovTrack	120K	1.1M	5	6

We analyze the performance of the algorithms with randomly generated importance queries. We created the queries by selecting random subgraphs of the data graph with n vertices and m edges. Random subgraphs are created by starting with a random vertex of the data graph. We iteratively add a randomly selected vertex from the neighborhood of any previously selected vertices. From the random subgraphs we created IQ-queries in the following way. We randomly selected c vertices of the subgraph, defining them as anchors, and mapped to the respective vertices of the data graph. The remaining $n - c$ vertices of the

randomly selected subgraph are defined as variables. The edges (including the edge labels) of the subgraph are edges in the query. With a probability p, a constraint is created from a numeric property of a vertex in the random subgraph. With a equal probability a constraint is a $>$ or $<$ constraint. The reference value of a $>$ constraint is the property value in the subgraph - 1, and the reference value of a $<$ constraint is the property value in the subgraph + 1. Scoring terms are created similarly to constraints. With a probability t, a numeric property of a vertex is select to be included in an IQ-term. If an IQ-term consists of more than one property, the properties are concatenated with a +. The aggregation function is MAX or SUM with equal probability.

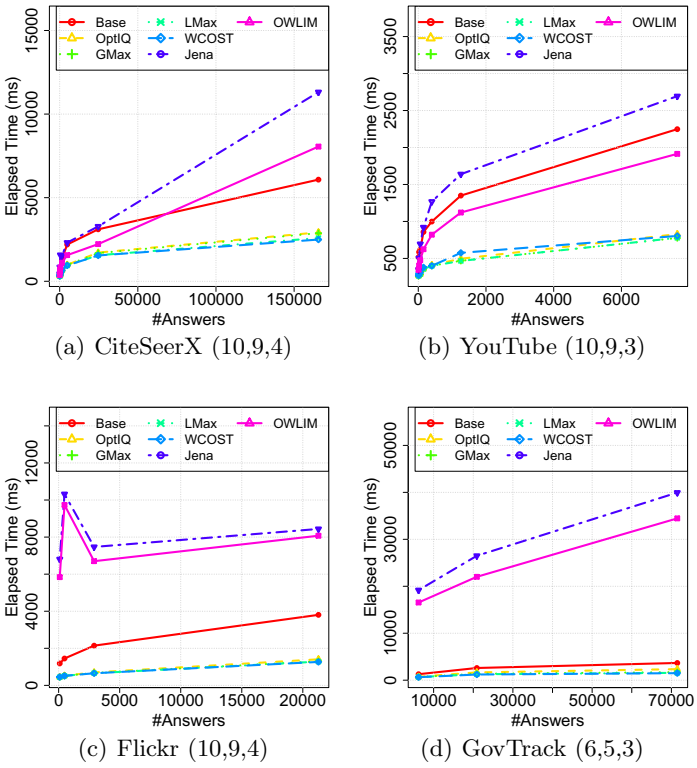
This query generation process ensures that all queries have at least one solution (which is the random subgraph the query has been generated from) and that (in probability) the distribution of structural patterns and properties used in constraints and query terms in a set of random queries resembles the respective distributions in the data graph.

5.2 Experimental Results

We evaluated our system by the selectivity of a query (i.e. the number of answers a query has), the size of the query (i.e. the number of vertices and edges the subgraph query has) and the number of desired answers. We used a set of 1000 random queries for the experiments.

Results by Selectivity. Figure 3 shows the runtime in relation to the answer size of the subgraph query. All algorithms show a sub-linear increase in the runtime with an increasing answer size. Reading subgraphs from disk is a dominating factor of the total runtime. The number of answers to the subgraph query increases much faster than the required number of subgraph reads because usually many answers lie in the same subgraphs. Compared to the baseline more sophisticated algorithms like WCOST and LMax can receive good speed-ups in some but not all settings - especially when the answer size is high. For non-selective queries our algorithms are up to one order of magnitude faster than the evaluated triple stores. For some datasets (Flickr, GovTrack) triple stores perform considerably worse even for very selective queries.

Results by Subgraph Query Size. For experiments on the subgraph query size, we use 2 pairs of query types (1000 random queries each) where each pair differs in the number of edges. The results of Figure 4 show that our algorithms scale very well in the number of vertices. As we increase the number of edges in a query, the runtime usually decreases as the query gets more selective (i.e. has fewer answers). Once again we see that our algorithms perform much better than the triple stores, and the performance difference is especially high for complex queries.



(a) CiteSeerX (10,9,4) (b) YouTube (10,9,3)

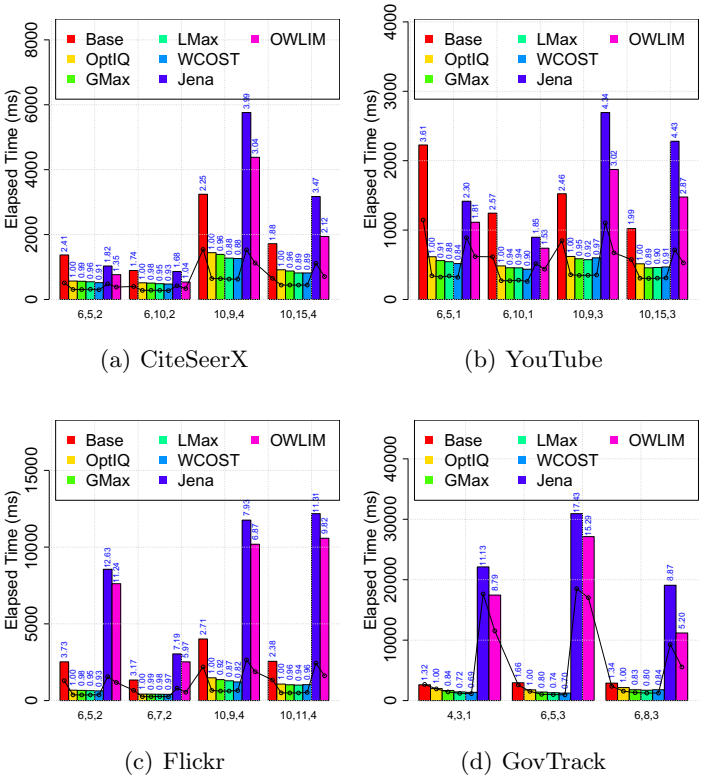
(c) Flickr (10,9,4) (d) GovTrack (6,5,3)

Fig. 3. Results by selectivity. Each caption shows the query size, e.g. 6,5,2 means 6 vertices, 5 edges, and 2 anchors.

Results by Parameter k. We analyzed the impact of the desired number of answers on runtime. Overall the scaling of our algorithms with respect to the number of answers is very good (see Figure 5), and is much less marked than for the triple stores. The almost constant runtime of our algorithms for low values of k is once again the result of the domination of the total runtime by the time needed to read a subgraph from disk. When most subgraphs in the neighborhood of the anchors have to be read to find the top-1 answer then the time to create a few additional solutions is low.

6 Related Work

We presented algorithms to identify the best answers to importance queries on attributed graphs. We extended subgraph matching algorithms to answer these queries, as non-specialist database systems (SQL as well as RDF databases) have a bad performance on complex subgraph queries. Subgraph queries on relational databases require many expensive self-joins on a potentially very large edge



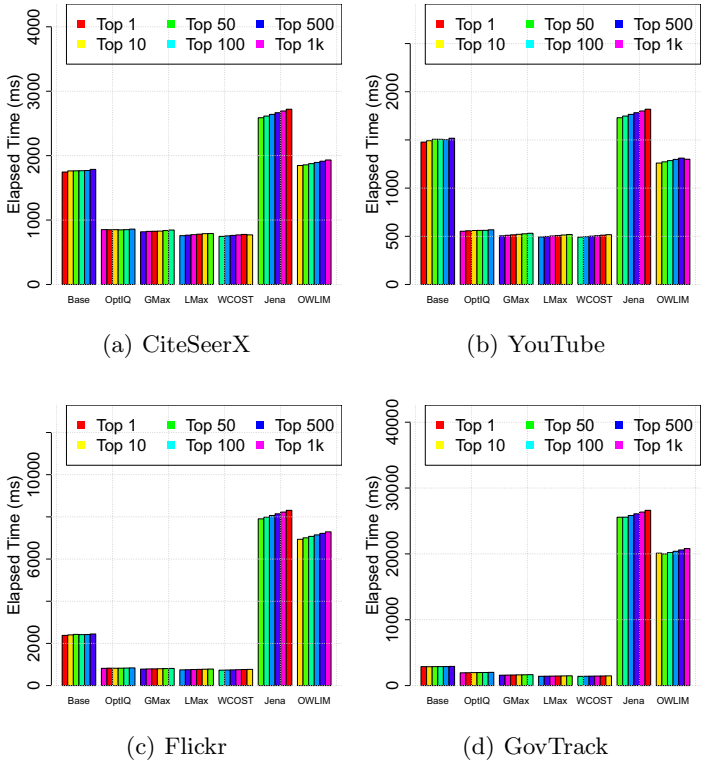
(a) CiteSeerX (b) YouTube

(c) Flickr (d) GovTrack

Fig. 4. Results by query size (Top 10, bar: mean, line: median)

table. RDF databases built on top of relational databases suffer from the same problem. Only RDF databases that store their data as graphs and not as triples could potentially provide a good performance. Recently Zou et al. [16] showed the performance advantage of subgraph matching algorithms compared to join-based algorithms used by triple stores for answering SPARQL queries. A way to improve the performance of triple stores for top-k SPARQL queries has been proposed by Magliacane et al. [9]. They presented a rank-aware join algorithm for top-k SPARQL queries. However, SPARQLRank supports only limited ranking functions and in particular does not support aggregation functions in the ranking term as we do.

So answering importance queries via subgraph matching algorithms is the best available approach, and with the Base algorithm we presented the straightforward way to answer importance queries by calling a subgraph matching algorithm. But we also showed that we can do much better than the simple Base approach by using sophisticated pruning techniques. Pruning strategies for subgraph matching have been discussed for decades. A considerable amount of literature has been published on subgraph matching on a single large graph. Since the early work of Ullman [12] most work on subgraph matching has been



(a) CiteSeerX (b) YouTube

(c) Flickr (d) GovTrack

Fig. 5. Results by parameter k of top-k queries

conducted on finding better ways to prune the search space of branch-and-bound algorithms. State-of-the-art algorithms store sophisticated graph invariants in precomputed indexes [4, 13, 14] to speed up the search. Invariants (in the simplest case the degree of a vertex) can be used to determine whether a vertex in the graph database *cannot* be a mapping for a variable in the query graph in an answer. Good overviews on different algorithms and pruning techniques are in [6, 11].

Our problem setting is different to this classical problem in two very important ways. First, classic subgraph matching searches only for structural patterns without anchors. This makes the overall computational effort much higher and is usually – depending on the dataset – in the range of hours. Second, the data graph is stored in memory. In our problem setting with anchored queries, the computational effort is much lower. However, our objective is to answer queries in interactive settings within seconds on large, disk-resident datasets. So I/O efficiency is an important issue for us. As we showed, we can answer most anchored queries in less than a second. This is less time than an in-memory algorithm spends loading the data graph into memory.

We defined importance queries as an extension of standard subgraph queries, but deriving it from approximate [15] or probabilistic [2] subgraph matching definitions is straightforward. Approximate matching algorithms do not search for exact matches, but for a subgraph similar to the query graph. Probabilistic matching algorithms work on probabilistic graphs, i.e. graphs that model the probability of the existence of an edge. The intent is to address the problem of errors in the data or limited knowledge of the system that is modeled in the graph. The techniques we developed for the advanced WCOST and GMAX algorithms could be transfered to related problems. Join-based algorithms for triple stores lack this flexibility.

7 Conclusion and Future Work

In this paper, we motivated and defined the problem of importance queries on graph databases. Such queries can also be expressed in SPARQL through the FILTER and ORDER BY constructs. We designed query algorithms for efficient retrieval of top-k answers to importance queries and evaluated the performance of the algorithms on large real-world.

By computing upper-bounds for the IQ-scores of partial substitutions, our most advanced algorithms are able to prune branches of the search tree that will not lead to a top-k answer. Thus, these algorithms achieve a significantly better performance than naive implementations. Our best algorithms need less than a second to answer the majority of our random test queries on graphs with up to about 15 million edges.

We believe importance queries are an important next step to personalized graph queries. As a next step, we plan to extend the concept to probabilistic subgraph matching. Then we can extend the search to probabilistic graph databases. For example, image probabilistic "acquaintance" edges in Facebook-style graph databases inferred from the co-occurrence of people in images.

Acknowledgements. We thank the anonymous reviewers and Barry Bishop and Nikolay Krustev from Ontotext for their helpful remarks. Parts of this work have been funded by the US Army Research Office under grant W911NF0910206.

References

[1] Apache Software Foundation: Apache Jena, `http://jena.apache.org`

[2] Bröcheler, M., Pugliese, A., Subrahmanian, V.S.: Probabilistic subgraph matching on huge social networks. In: 2011 International Conference on Advances in Social Networks Analysis and Mining (ASONAM), pp. 271–278 (2011)

[3] Bröcheler, M., Pugliese, A., Subrahmanian, V.S.: DOGMA: A disk-oriented graph matching algorithm for RDF databases. In: Bernstein, A., Karger, D.R., Heath, T., Feigenbaum, L., Maynard, D., Motta, E., Thirunarayan, K. (eds.) ISWC 2009. LNCS, vol. 5823, pp. 97–113. Springer, Heidelberg (2009)

[4] Cordella, L., Foggia, P., Sansone, C., Vento, M.: A (sub)graph isomorphism algorithm for matching large graphs. IEEE Transactions on Pattern Analysis and Machine Intelligence 26(10), 1367–1372 (2004)
[5] Karypis, G.,, V.K.: A fast and high quality multilevel scheme for partitioning irregular graphs. SIAM Journal on Scientific Computing 20(1), 359–392 (1998)
[6] Gallagher, B.: Matching structure and semantics: A survey on graph-based pattern matching. In: 2006 AAAI Fall Symposium on Capturing and Using Patterns for Evidence Detection, pp. 45–53 (2006)
[7] Hendrickson, B., Leland, R.: A multi-level algorithm for partitioning graphs. In: Proceedings of the 1995 ACM/IEEE Conference on Supercomputing, p. 28 (1995)
[8] Kiryakov, A., Ognyanov, D., Manov, D.: OWLIM - a pragmatic semantic repository for OWL. In: Dean, M., Guo, Y., Jun, W., Kaschek, R., Krishnaswamy, S., Pan, Z., Sheng, Q.Z. (eds.) WISE 2005 Workshops. LNCS, vol. 3807, pp. 182–192. Springer, Heidelberg (2005)
[9] Magliacane, S., Bozzon, A., Della Valle, E.: Efficient execution of top-k sparql queries. In: Cudré-Mauroux, P., Heflin, J., Sirin, E., Tudorache, T., Euzenat, J., Hauswirth, M., Parreira, J.X., Hendler, J., Schreiber, G., Bernstein, A., Blomqvist, E. (eds.) ISWC 2012, Part I. LNCS, vol. 7649, pp. 344–360. Springer, Heidelberg (2012)
[10] Neumann, T., Weikum, G.: Rdf-3x: a risc-style engine for rdf. Proc. VLDB Endow. 1(1), 647–659 (2008)
[11] Solnon, C.: All different-based filtering for subgraph isomorphism. Artificial Intelligence 174(12-13), 850–864 (2010)
[12] Ullmann, J.R.: An algorithm for subgraph isomorphism. Journal of the ACM 23(1), 31–42 (1976)
[13] Washio, T., Motoda, H.: State of the art of graph-based data mining. SIGKDD Exploration Newsletter 5(1), 59–68 (2003)
[14] Zhang, S., Li, S., Yang, J.: Gaddi: distance index based subgraph matching in biological networks. In: Proceedings of the 12th International Conference on Extending Database Technology, EDBT 2009, pp. 192–203. ACM (2009)
[15] Zhang, S., Yang, J., Jin, W.: Sapper: subgraph indexing and approximate matching in large graphs. Proc. VLDB Endow. 3(1-2), 1185–1194 (2010)
[16] Zou, L., Mo, J., Chen, L., Özsu, M.T., Zhao, D.: gStore: answering SPARQL queries via subgraph matching. Proc. VLDB Endow. 4(8), 482–493 (2011)

Towards an Automatic Creation of Localized Versions of DBpedia

Alessio Palmero Aprosio[1], Claudio Giuliano[2], and Alberto Lavelli[2]

[1] Università degli Studi di Milano, via Comelico 39/41, 20135 Milano, Italy
alessio.palmero@unimi.it
[2] Fondazione Bruno Kessler, Via Sommarive 18, 38123 Trento, Italy
{giuliano,lavelli}@fbk.eu

Abstract. DBpedia is a large-scale knowledge base that exploits Wikipedia as primary data source. The extraction procedure requires to manually map Wikipedia infoboxes into the DBpedia ontology. Thanks to crowdsourcing, a large number of infoboxes has been mapped in the English DBpedia. Consequently, the same procedure has been applied to other languages to create the localized versions of DBpedia. However, the number of accomplished mappings is still small and limited to most frequent infoboxes. Furthermore, mappings need maintenance due to the constant and quick changes of Wikipedia articles. In this paper, we focus on the problem of automatically mapping infobox attributes to properties into the DBpedia ontology for extending the coverage of the existing localized versions or building from scratch versions for languages not covered in the current version. The evaluation has been performed on the Italian mappings. We compared our results with the current mappings on a random sample re-annotated by the authors. We report results comparable to the ones obtained by a human annotator in term of precision, but our approach leads to a significant improvement in recall and speed. Specifically, we mapped 45,978 Wikipedia infobox attributes to DBpedia properties in 14 different languages for which mappings were not yet available. The resource is made available in an open format.

1 Introduction

DBpedia is a community project[1] aiming to develop a large-scale knowledge base that exploits Wikipedia as primary data source. Wikipedia represents a practical choice as it is freely available under Creative Commons License, covers an extremely large part of human knowledge in different languages (45 out of 285 have more than 100,000 articles), and is populated by more than 100,000 active contributors, ensuring that the information contained is constantly updated and verified. At the time of starting this paper, the English DBpedia contained about 3.77 million entities, out of which 2.35 millions are classified in the DBpedia Ontology, available as Linked Data,[2] and via DBpedia's main SPARQL endpoint.[3] Due to the large and constantly increasing number

[1] http://dbpedia.org/
[2] http://wiki.dbpedia.org/Downloads
[3] http://dbpedia.org/sparql

H. Alani et al. (Eds.): ISWC 2013, Part I, LNCS 8218, pp. 494–509, 2013.

of links from and to other data sources, DBpedia continues to gain popularity and today it plays a central role in the development of the Web of Data.

The DBpedia ontology, consisting of 359 classes (e.g., person, city, organization) – organized in a subsumption hierarchy – and 1,775 properties (e.g., birth place, latitude, family name), is populated using a semi-automatic rule-based approach that relies prominently on Wikipedia *infoboxes*, a set of attribute-value pairs that represent a summary of the most important characteristics Wikipedia articles have in common. For example, country pages in the English Wikipedia typically contain the infobox `Infobox_country` containing specific attributes such as `currency`, `population`, `area`, etc. Specifically, the DBpedia project provides an information extraction framework[4] used, first, to extract the structured information contained in the infoboxes and, second, to convert it into RDF triples. Then, crowdsourcing is extensively used to map infoboxes and their attributes to the classes and properties of the DBpedia Ontology, respectively. For example, the `Infobox_country` is mapped to the class `Country` and its attribute `area` is mapped to the property `areaTotal`. Finally, all Wikipedia articles (instances) containing mapped infoboxes are automatically added to the DBpedia ontology, and mapped properties are used to add facts (statements) describing these instances. There are three main problems to solve. First, infoboxes do not have a common vocabulary, as the collaborative nature of Wikipedia leads to a proliferation of variants for the same concept. This problem is addressed using crowdsourcing, a public wiki for writing infobox mappings: editing existing ones, as well as editing the ontology, is available since DBpedia 3.5. Second, the number of infoboxes is very large, and consequently the mapping process is time consuming. To mitigate this problem, the mapping process follows an approach based on the frequency of infobox usage in Wikipedia articles. Most frequent elements are mapped first, ensuring a good coverage as infobox utilization follows the Zipf's distribution [17]. In this way, even though the number of mappings is small, a large number of Wikipedia articles can be added to the knowledge base. Third, mappings need maintenance due to the constant and quick changes of Wikipedia articles. For example, the Italian template `Cardinale_della_chiesa_cattolica` (Cardinal of the Catholic Church) has been replaced by a more generic `Cardinale` (Cardinal). In this particular case, the Wikipedia editors decided to delete the template, without creating a redirect link, therefore the mapping[5] between the template and the DBpedia class `Cardinal` becomes orphan, and the DBpedia extraction framework is no longer able to extract the corresponding entities.

At the early stages of the project, the construction of DBpedia was solely based on the English Wikipedia. More recently, other contributors around the world have joined the project to create localized and interconnected versions of the knowledge base. The goal is to populate the same ontology used in the English project, extracting articles from editions of Wikipedia in different languages. In its current version 3.8, DBpedia contains 16 different localized datasets and the information extraction framework has been extended to provide internationalization and multilingual support [7].

[4] `http://dbpedia.org/documentation`

[5] `http://mappings.dbpedia.org/index.php/Mapping_it:Cardinale_della_chiesa_cattolica`

However, the inclusion of more languages has emphasized the problems described above. Furthermore, the DBpedia ontology needs frequent extensions and modifications as it has been created on the English Wikipedia, while each edition of Wikipedia is managed by different groups of volunteers with different guidelines.

In this paper, we focus on the problem of automatically mapping infobox attributes to properties into the DBpedia ontology for extending the coverage of the existing localized versions (e.g., Italian, Spanish) or building from scratch versions for languages not yet covered (e.g., Swedish, Norwegian, Ukranian). This task is currently performed using crowdsourcing and there are no published attempts to perform it automatically. Related work has exclusively focused on developing automatic approaches to attribute mapping between different Wikipedia editions; these results can be used to automatize the mapping process, though this solution is highly prone to changes in Wikipedia, a noticeable drawback considering how fast edits are made. This study is complementary to previous investigations in which we studied the mapping of infoboxes to classes in the DBpedia ontology [10,11]. The above problem can be classified as schema matching, limited to alignment as we do not perform any successive merging or trasforming.

We propose an instance-based approach, that exploits the redundancy of Wikipedia in different editions (languages), assuming that attributes and properties are equivalent if their values are similar. Specifically, the mapping is cast as a binary classification task in which instances are infobox attribute/ontology property pairs extracted from versions of Wikipedia and DBpedia in different languages and cross-language links are used to represent the instances in a unified space. This allows us to learn the mapping function, for example, from existing mappings in English and German and predict Swedish instances. Attributes and properties are compared using their values taking into account their types (i.e., date, integer, object, etc.). For attributes, the type is calculated; for properties, the type is given by the ontology. We show that this approach is robust with respect to rapid changes in Wikipedia, differently from approaches that first map infoboxes among Wikipedia editions. The evaluation has been performed on the Italian mappings. We compared our results with the current mappings on a random sample re-annotated by the authors. We report results comparable to the ones obtained by a human annotator in term of precision (around 87%), but our approach leads to a significant improvement in recall (around 80%) and speed.

Finally, we mapped 45,978 Wikipedia infobox attributes to DBpedia properties in 14 different languages for which mappings were not yet available; the resource is made available in an open format.[6]

2 Problem Formalization

We consider the problem of automatically mapping attributes of Wikipedia infoboxes into properties of the DBpedia ontology. The problem can be classified as schema/ontology matching in which we are interested in equivalence relations between attributes and properties.

An infobox is a set of *attribute/value* pairs that represent a summary of the most salient characteristics Wikipedia articles have in common. For example, the

[6] `http://www.airpedia.org/`

infobox `Officeholder` in the English Wikipedia contains generic attributes, such as `name`, `birth_date`, and `birth_place`, and specific ones, such as `term_start`, `party`, and `office`. Notice that each Wikipedia edition is maintained by different communities and has different guidelines that can have a strong impact on the mapping results. For example, in the Italian edition, `Carica_pubblica` (`Officeholder`) does not contain generic attributes that are usually contained in the infobox `Bio`. In addition, there are no constraints on types, therefore in some editions of Wikipedia there can be a single attribute `born` containing both place and date of birth, while other languages decide to split this information into different attributes.

A DBpedia property is a relation that describes a particular characteristic of an object. It has a *domain* and a *range*. The domain is the set of objects where such property can be applied. For instance, `birthDate` is a property of `Person`, therefore `Person` is its domain. Around 20% of the DBpedia properties use the class `owl:Thing` as domain. The range is the set of possible values of the property. It can be a scalar (date, integer, etc.) or an object (`Person`, `Place`, etc.). For example, the range of `birthDate` is date and the range of `spouse` is `Person`.

Manual mappings are performed as follows. First, human annotators assign an infobox to a class in the DBpedia ontology. Then, they map the attributes of the infobox to the properties of the ontology class (or to its ancestors). An example of mapping is shown in Figure 1.

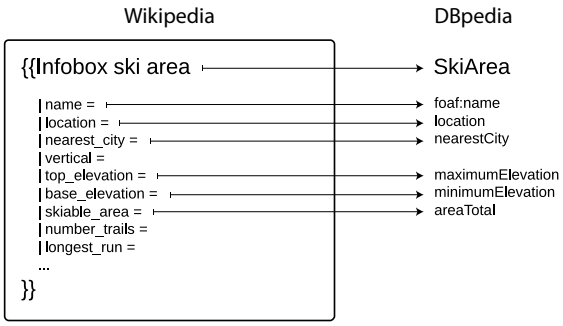


Fig. 1. Example of DBpedia mapping

The rest of the section is devoted to analyze the difficulties to adapt existing systems that perform infobox matching and completion (e.g., [13,4,1]) to solve this task. We could use existing approaches to map infoboxes between different Wikipedia editions and, then, use the existing DBpedia mappings to extend the mappings to languages not yet covered. An example is shown in Figure 2, where the template `Persondata` in English has been mapped to `Bio` in Italian, and similarly `Officeholder` to `Carica_pubblica`. Suppose that Italian mappings do not exist yet, they can be derived using the existing English DBpedia mappings. However, approaching the problem in this manner leads to a series of problems.

– Alignment of Wikipedia templates in different languages is often not possible, because there are no shared rules among the different Wikipedia communities on the management of infoboxes. In the example of Figure 2, `Carica_pubblica` only refers to politician, while `Officeholder` is more general.
– Properties may be mapped to different infoboxes in different languages. For example, the Italian DBpedia uses attributes of the `Bio` template to map generic biographical information, because specialized templates, such as `Carica_pubblica`, in the Italian Wikipedia do not contain generic information. This is not true in the English edition and in many other languages.
– Due to the previous point, some infoboxes are not mapped to any DBpedia class. This is the case of the `Persondata` template in English: since its information is repeated in the more specialized templates (for example, date of birth, name, occupation), the DBpedia annotators ignored it. A system that should align `Bio` and `Persondata`, and then transfer the mappings from English to Italian, would not map `Bio` to any DBpedia class since there is no mapping available for `Persondata`; therefore, all the generic biographical information would be lost.

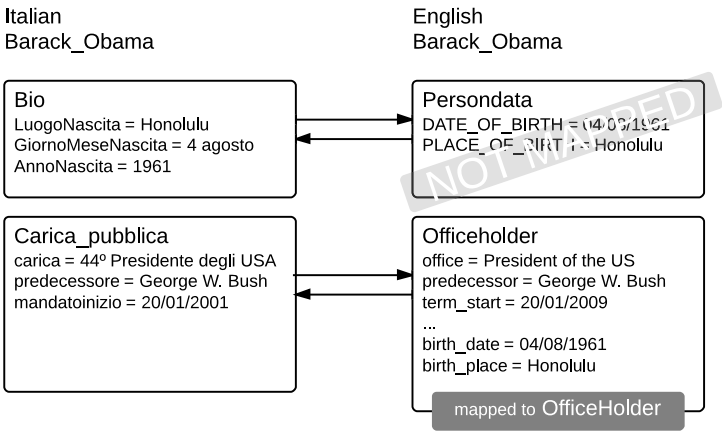


Fig. 2. An example of infobox alignment

3 Workflow of the System

In this work, we propose an automatic system for generating DBpedia mappings. Formally, given an infobox I and an attribute A_I contained in I, our system maps the pair $\langle I, A_I \rangle$ to a relation R in the DBpedia ontology.

Our approach exploits the redundancy of Wikipedia across editions in different languages, assuming that, if values of a particular infobox attribute are similar to values of a particular DBpedia property, then we can map the attribute to the property.

This approach requires existing versions of DBpedia to train the system, in particular we exploit the English, German, French, Spanish, and Portuguese editions. Given a target language l, the system extracts the mappings between DBpedia properties and infobox atttributes in such language. Note that the target language l can also be included in the set of languages chosen as training data; however, in our experiments we do not use this approach since we are interested in building mappings for those chapters of Wikipedia for which the corresponding DBpedia does not exist yet. Our system consists of three main modules: pre-processing, mapping extraction, and post-processing. Figure 3 depicts the workflow of the system.

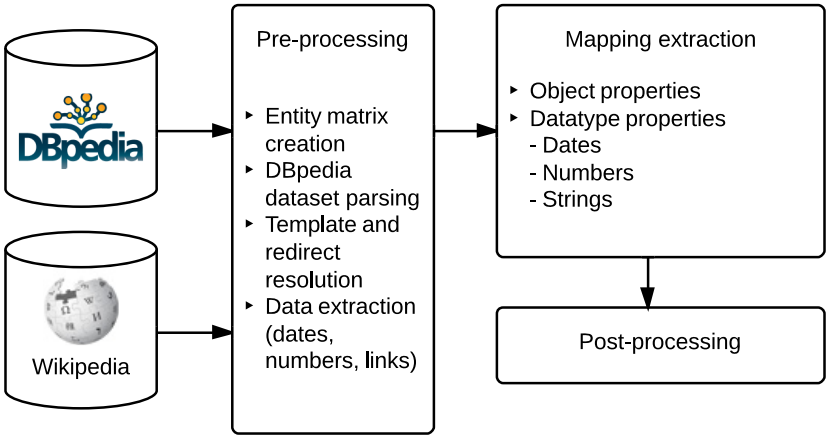


Fig. 3. Workflow of the system

4 Pre-processing

This section describes how we collect and normalize the data needed for the mapping between DBpedia and Wikipedia.

4.1 Entity Matrix Creation

The proposed approach makes considerable use of the redundancy of information among different versions of Wikipedia. In particular, we focus on the semi-structured information contained in the infoboxes. For example, the English Wikipedia page of Barack Obama contains an infobox with his birth date, birth place, etc. The same information is often included in the infoboxes of the corresponding pages in other Wikipedia editions. Therefore, the first step consists in building a matrix that aggregates the entities

(rows) in the different languages of Wikipedia (columns). The alignment is trivial as Wikipedia provides cross-language links between pairs of articles describing the same concept in different editions.

The accuracy of cross-language links has been investigated in the Semantic Web community [13,7], and conflicts have been found in less than 1% of the articles. In our implementation, when a conflict is found, the corresponding page is discarded.

In the rest of the paper, $P_{l_1}, P_{l_2}, \ldots$ denote the Wikipedia pages in languages $l_1, l_1, \ldots$, and P denotes the entity described by the corresponding row in the entity matrix. Figure 4 shows a portion of the matrix.

en	de	it	es	...
Xolile Yawa	Xolile Yawa	*null*	*null*	...
The Locket	*null*	Il segreto del medaglione	*null*	...
Barack Obama	Barack Obama	Barack Obama	Barack Obama	...
null	*null*	Giorgio Dendi	*null*	...
Secoya People	*null*	Secoya	Aido pai	...
...	...	...	...	...

Fig. 4. A portion of the entity matrix

4.2 DBpedia Dataset Parsing

DBpedia releases its ontology description in OWL format. The source file contains the description of the classes and properties, with all their characteristics. In our case, we search for the type (range) of each property. Depending on this feature, we can split them into two categories:

- *Datatype properties*, when the relation connects instances of classes to literals of XML (scalar values). For example `birthDate` connects a `Person` to a date.
- *Object properties*, when the relation connects instances of two classes (not necessarily different). For example, `birthPlace` connects a `Person` to a `Place` and `spouse` connects a `Person` to a `Person`.

Performing the mapping task, we use different strategies depending on the range of the category.

4.3 Template and Redirect Resolution

In Wikipedia, templates are particular pages created to be included into other pages. Infoboxes are a particular subset of templates that are usually rendered as a table in the upper-right corner of the corresponding Wikipedia article. Although this particular subset of templates is useful for information extraction from Wikipedia, only around 10% of templates belong to this category: the majority of them is used to give graphic

coherence to the same types of elements in different articles. For example, countries are often shown in Wikipedia infoboxes as the flag of the country followed by the name. These templates are often used as values for the infobox attributes. Since different languages have different strategies in using templates, the alignment between values containing templates is not trivial. During the alignment phase, these discrepancies may lead to errors. To address this problem, we pre-process the attribute values using the *Bliki engine*,[7] a parser that converts templates to their expanded text. After this operation, templates such as `{{EGY}}` are rendered as the Egypt flag followed by the name of the country linked to its page.

4.4 Data Extraction

In our approach, the main difficulty consists in the comparison between data obtained from DBpedia and attribute values stored in Wikipedia infoboxes. This is due to the fact that DBpedia is strongly typed, while Wikipedia does not have an explicit type system. Attribute values often contain a mixture of dates, numbers, and text, represented, formatted, and approximated in different ways depending on the Wikipedia edition and on the users who edit articles. These types of data can be formatted in different ways in different languages. For example, in English, we can express a date using different patterns, such as, "June 4th, 1983", "04/06/1983", or even "06/04/1983." Furthermore, numeric values can be approximated using variable precision depending on a particular edition of Wikipedia. For instance, the total area of Egypt is 1,002,450 in the English Wikipedia and 1.001.449 in the Italian one, where both the value and the format are different.

To tackle these problems, we defined a function e that, using a set of heuristics for numbers and dates, extracts – for each attribute value – four different sets of elements: numbers, dates, links and text tokens.

attribute	value
name	Diego Maradona
image	Maradona at 2012 GCC Champions League final.JPG
image_size	250
birth_place	[[Lanús]], [[Buenos Aires province|Buenos Aires]], [[Argentina]]
birth_date	{{Birth date and age|1960|10|30|df=yes}}
height	{{height|m=1.65}}
youthyears1	1968–1969
youthyears2	1970–1974
youthyears3	1975–1976
...	...

Fig. 5. `Infobox_football_biography` attributes for Diego Maradona

[7] `https://code.google.com/p/gwtwiki/`

In Figure 5 an example of `Infobox_football_biography` is presented. In the birth_place value, the value "[[Lanús]], [[Buenos Aires province|Buenos Aires]], [[Argentina]]" of the attribute *birth_place* is converted into the bag of links {`Lanús`, `Buenos_Aires_province`, `Argentina`} and the set of tokens {Lanús, ",", Buenos, Aires, ",", Argentina}, leaving the remaining sets (dates and numbers) empty. In the birth_date value, the template "Birth date and age" is parsed using the Bliki engine (see Section 4.3), resulting in "30 October 1960 (age 52)"; then, the string is converted into the set of dates {1960-10-30}, the set of numbers {30, 1960, 52}, and the set of tokens {30, October, 1960, (, age, 52,)}, leaving the links set empty.

5 Mapping Extraction

In this section, we describe the matching algorithm used to determine whether an attribute A_I contained in the infobox I in Wikipedia can be mapped to a given property R in DBpedia. To find the mappings, we have to calculate the pairwise similarities between the elements in the set of all the possible attributes A_I and the elements in the set of all the possible properties R. The candidates are represented as pairs (A_I, R), the pairs with the highest similarity $S(A_I, R)$ are considered correct mappings. The similarity is an average result calculated using instance-based similarities between the values of property R in different DBpedia editions and the values of the attribute A_I in different Wikipedia pages in the target language. This process can lead to large number of comparisons to determine if a pair (A_I, R) can be mapped. The rest of the section provides a detailed and formal description of the algorithm.

Given a relation R in DBpedia in languages $L = \{l_1, l_2, \ldots, l_n\}$ and a target language l, the algorithm works as follows.

1. We build the following set, discarding entities that are not involved in the relation:

$$\Pi_R = \{P_{l_i} : P_{l_i} \text{ has its corresponding } P_l \text{ and exists at least an instance of } R \text{ in DBpedia in language } l_i.\}$$

2. For each pair (A_I, R), we compute S_l:

$$S_l(A_I, R) = \frac{\sum_{P_{l_i} \in \Pi_R} \sigma_l(e(A_I, P_l), v(R, P_{l_i}))}{|\Pi_R|}$$

   where the function σ_l is defined in Section 6 and the division by $|\Pi_R|$ is used to calculate the average similarity between attributes and properties based on their values in different languages.
3. All pairs A_I, R for which $S_l(A_I, R) < \lambda$ are discarded. Varying λ, we can change the trade-off between precision and recall.
4. For each infobox I, for which at least a pair (A_I, R) exists, we select A_I^* such that the pair (A_I^*, R) maximizes the function S.
5. Finally, we obtain the set M_R of the selected pairs (A_I, R).

6 Inner Similarity Function

The inner similarity $\sigma_l(e(A_I, P_l), v(R, P_{l_i})) \rightarrow [0, 1]$ is computed between the value of A_I in language l, extracted and normalized by the function e defined in Section 4.4, and the values of R in the DBpedia editions in languages $l_1, l_2, \ldots, l_n$, extracted by the function v. In sections 6.1 and 6.2, the function σ_l is formally defined depending on the two categories used to classify the property R (see Section 4.2). We use V_W and V_D to indicate the values returned by the functions e and v, respectively.

6.1 Similarity between Object Properties

When the range of the property R is an object, the value V_D corresponds to a Wikipedia page. Using the entity matrix E, we look for the equivalent page V_D^l in the target language l. Then, we search V_D^l in the links set of V_W, and we set $\sigma_l(V_D, V_W) = 1/k$ if we find it – k is the cardinality of the links subset of V_W. By dividing by k, we downgrade the similarity in case of partial matching. If the links set of V_W does not contain V_D^l, or if V_D does not have a corresponding article in the target language (and therefore V_D^l does not exist), we compare the string representations of V_D and V_W (see Section 6.2).

6.2 Similarity between Datatype Properties

When the range of the property R is not an object, we handle 9 types of data: calendar related (*date*, *gYearMonth*, *gYear*), numeric (*double*, *float*, *nonNegativeInteger*, *positiveInteger*, *integer*), and *string*. We discard the `boolean` type, as it affects only 4 properties out of 1,775, and it is never used in languages different from English.

Calendar Related Data. Given the value V_D of type *date* and the set V_W, we compute $\sigma_l(V_D, V_W)$ by searching the day, the month and the year of V_D in the set V_W. In particular, the month is given only if it appears as text, or if it is included in the numbers set of V_W together with the day and the year. Similarly, we look at the day only if it appears with the month. We look at the date parts separately, because some Wikipedia editions split them into different infobox attributes. We assign a value of $1/3$ to each part of the date V_D that appears in V_W.

$$\sigma_l(V_D, V_W) = \begin{cases} 1 & \text{if day-month-year are present in } V_W \\ 2/3 & \text{if day-month are present in } V_W \\ 2/3 & \text{if month-year are present in } V_W \\ 1/3 & \text{if year is present in } V_W \end{cases}$$

Similarly, for *gYearMonth* we set $\sigma_l(V_D, V_W) = 1$ if both month and year appear in the dates set of V_W, and $\sigma_l(V_D, V_W) = 0.5$ if V_W contains only one of them. Finally, for *gYear* we set $\sigma_l(V_D, V_W) = 1$ if the year is included in the numbers set of V_W.

Numeric Data. While for calendar related data we expect to find the exact value, often properties involving numbers can have slightly different values in different languages (see Section 4.4 for an example). If $V_D = 0$, we check if the numbers subset of V_W contains 0. If true, then $\sigma_l(V_D, V_W) = 1$, otherwise $\sigma_l(V_D, V_W) = 0$. If $V_D \neq 0$, we search for values in V_W near to V_D, setting a tolerance $\nu > 0$. For each n in the numbers set of V_W, we calculate $\varepsilon = |V_D - n| \,/\, |V_D|$. If $\varepsilon < \nu$, then we set $\sigma_l(V_D, V_W) = 1$ and exit the loop. If the end of the loop is reached, we set $\sigma_l(V_D, V_W) = 0$.

Strings. String kernels are used to compare strings. To compute the similarity, this family of kernel functions takes into account two strings and looks for contiguous and non-contiguous subsequences of a given length they have in common. Non contiguous occurrences are penalized according to the number of gaps they contain. Formally, let Σ be an alphabet of $|\Sigma|$ symbols, and $s = s_1 s_2 \dots s_{|s|}$ a finite sequence over Σ (i.e., $s_i \in \Sigma, 1 \leqslant i \leqslant |s|$). Let $\mathbf{i} = [i_1, i_2, \dots, i_n]$, with $1 \leqslant i_1 < i_2 < \dots < i_n \leqslant |s|$, be a subset of the indices in s, we will denote as $s[\mathbf{i}] \in \Sigma^n$ the subsequence $s_{i_1} s_{i_2} \dots s_{i_n}$. Note that $s[\mathbf{i}]$ does not necessarily form a contiguous n-gram of s. The length spanned by $s[\mathbf{i}]$ in s is $l(\mathbf{i}) = i_n - i_1 + 1$. The gap-weighted subsequences kernel (or string kernel) of length n is defined as

$$K_n(s,t) = \langle \phi^n(s), \phi^n(t) \rangle = \sum_{u \in \Sigma^n} \phi^n_u(s) \phi^n_u(t), \tag{1}$$

where

$$\phi^n_u(s) = \sum_{\mathbf{i}: u = s[\mathbf{i}]} \mu^{l(\mathbf{i})}, u \in \Sigma^n \tag{2}$$

and $\mu \in]0, 1]$ is the decay factor used to penalize non-contiguous subsequences.[8] An explicit computation of Equation 1 is unfeasible even for small values of n. To evaluate more efficiently K_n, we use the recursive formulation based on a dynamic programming implementation [8,14,5].

In our implementation, subsequences are n-grams (strings are tokenized), where $n = \min\{|V_D|, |V^*_W|\}$ and V^*_W is the tokenized set of V_W where some n-grams have been replaced with their translation when cross-language links exist. The similarity function is defined as the first strictly positive value returned by the following loop:

$$\sigma_l(V_D, V_W) = \frac{K_i(V_D, V^*_W)}{n - i + 1} \qquad \text{for each } i = n, n-1, \dots, 1.$$

7 Post-processing

Some infoboxes contain attributes with multiple values. For example, the musical genre of a particular album can be "rock" and "pop", or a book can have more than one author. In these cases, Wikipedia provides more than one attribute describing the same relation, and adds an incremental index after the name of the attribute (sometimes

[8] Notice that by choosing $\mu = 1$ sparse subsequences are not penalized. The algorithm does not take into account sparse subsequences with $\mu \to 0$.

also adding an underscore between the attribute name and the index). For example, the `Infobox_settlement` template contain the attribute `twinX` used for twin cities, where X can vary from 1 to 9. In our system, if M_R contains a mapping $A_I \rightarrow R$, we also add the set of mappings $A'_I \rightarrow R$ where the name of attribute A' differs from A only for an added or replaced digit. This filter is applied on the set M of mappings built in the mapping phase (Section 5) and is only used to increase recall.

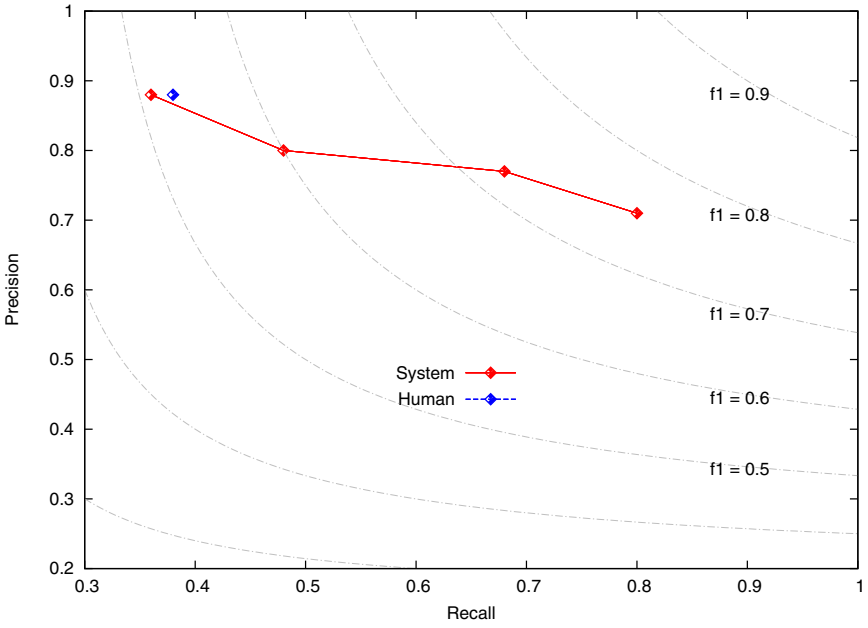


Fig. 6. Precision/recall curve of our system compared with the DBpedia original manual mapping in Italian. From left to right, λ value is 0.9, 0.7, 0.5, and 0.3.

8 Evaluation

Experiments have been carried on Italian, using existing DBpedia editions in five languages (English, Spanish, Portuguese, German, and French) as training data. To perform the evaluation, three annotators created a gold standard by manually annotating 15 infoboxes (for a total of 100 different attributes), randomly extracted from the first 100 most frequent infoboxes in the Italian Wikipedia. The inter annotator agreement is 91%, with respect to Fleiss' kappa measure [6]. The gold standard is available online on the Airpedia website.[9] As baseline, we use the manually mapped Italian infoboxes that can be downloaded from the DBpedia official website. [10] Specifically, we used the

[9] `http://www.airpedia.org/download/dbpedia-property-mappings-in-14-languages/`

[10] `http://mappings.dbpedia.org/`

version available on April 5th, 2013, made available by the Italian DBpedia project,[11] consisting of around 50 infoboxes and 469 attributes (in 18 infoboxes) mapped by one annotator during the spring 2012.

Figure 6 shows the precision/recall curve. Different precision/recall points are obtained by varying the parameter λ described in Section 5. The grey dashed lines join points with the same F_1. The results show that the coverage of the baseline (Human) is around 38% with a precision of around 88%. Our system is able to achieve comparable results in term of precision (87%), but it leads to a significant improvement in recall maintaining acceptable precision. Specifically, we can see that, by exploiting existing mappings, we can cover up to 70% of the attributes with a precision around 80%. Even though the procedure is not generally error-prone, we believe that it can be used as a starting point for releasing new DBpedia editions or extending existing ones. In the next section, we describe the current release of the resource.

9 The Resource

Overall, our system mapped 45,978 Wikipedia infobox attributes to DBpedia properties in 14 different languages for which mappings do not yet exist.[12] For each language, we only consider templates that appear more than 10 times in the corresponding Wikipedia and release the mappings paired with the value of the function f, described in the Section 5. The system has been trained on the DBpedia datasets in 6 languages (English, Italian, French, German, Spanish and Portuguese).

Table 1 shows the number of mappings extracted for each language ($\lambda = 0.3$). Notice that, even if the precision is not 100% and the process still needs human supervision, our approach can drastically reduce the time required, estimated in around 5 minutes per mapping per language if performed from scratch.[13]

Table 1. Mappings extracted and available as a resource

Language	Mappings	Language	Mappings
Belarusian	1,895	Norwegian	4,226
Danish	3,303	Romanian	4,563
Estonian	1,297	Slovak	2,407
Finnish	3,766	Albanian	1,144
Icelandic	646	Serbian	4,343
Lithuanian	3,733	Swedish	5,073
Latvian	2,085	Ukranian	5,760

10 Related Work

The main reference for our work is the DBpedia project [2]. Started in 2007, it aims at building a large-scale knowledge base semi-automatically extracted from Wikipedia.

[11] `http://it.dbpedia.org/`

[12] The complete resource is available at `http://www.airpedia.org/`

[13] This is an average time evaluated during the mapping of the Italian DBpedia.

Wikipedia infobox attribute names do not use the same vocabulary, and this results in multiple properties having the same meaning but different names and vice versa. In order to do the *mapping-based* extraction, DBpedia organizes the infobox templates into a hierarchy, thus creating the DBpedia ontology with infobox templates as classes. They manually construct a set of property and object extraction rules based on the infobox class. Nowadays, the ontology covers 359 classes which form a subsumption hierarchy and are described by 1,775 different properties. The English version is populated by around 1.7M Wikipedia pages, although the English Wikipedia contains almost 4M pages.

Yago [16], similarly to DBpedia, extracts structured information and facts from Wikipedia using rules on page categories. Conversely, FreeBase [3] and WikiData [18] are collaborative knowledge bases composed mainly by their community members.

The problem faced in this paper falls into the broader area of schema matching. A general survey on this topic is presented by Rahm and Bernstein [12]. Their work compares and describes different techniques, establishing also a taxonomy that is used to classify schema matching approaches. Similarly, Shvaiko and Euzenat [15] present a new classification of schema-based matching techniques. It also overviews some of the recent schema/ontology matching systems, pointing which part of the solution space they cover.

Bouma et al. [4] propose a method for automatically completing Wikipedia templates. Cross-language links are used to add and complete templates and infoboxes in Dutch with information derived from the English Wikipedia. First, the authors show that alignment between English and Dutch Wikipedia is accurate, and that the result can be used to expand the number of template attribute-value pairs in Dutch Wikipedia by 50%. Second, they show that matching template tuples can be found automatically, and that an accurate set of matching template/attribute pairs can be derived using intersective bidirectional alignment. In addition, the alignment provides valuable information for normalization of template and attribute names and can be used to detect potential mistakes. The method extends the number of tuples by 50% (27% for existing Dutch pages).

Adar et al. [1] present Ziggurat, an automatic system for aligning Wikipedia infoboxes, creating new infoboxes as necessary, filling in missing information, and detecting inconsistencies between parallel articles. Ziggurat uses self-supervised learning to allow the content in one language to benefit from parallel content in others. Experiments demonstrate the method's feasibility, even in the absence of dictionaries.

Nguyen et al. [9] propose WikiMatch, an approach for the infobox alignment task that uses different sources of similarity. The evaluation is provided on a subset of Wikipedia infoboxes in English, Portuguese and Vietnamese.

More recently, Rinser et al. [13] propose a three-stage general approach to infobox alignment between different versions of Wikipedia in different languages. First, it aligns entities using inter-language links; then, it uses an instance-based approach to match infoboxes in different languages; finally, it aligns infobox attributes, again using an instance-based approach.

11 Conclusion and Future Work

In this paper, we have studied the problem of automatically mapping the attributes of Wikipedia infoboxes to properties of the DBpedia ontology. To solve this problem, we have devised an instance-based approach that uses existing DBpedia editions as training data. We evaluated the system on Italian data, using 100 manually annotated infobox attributes, demonstrating that our results are comparable with the current mappings in term of precision (87% versus 88% for the human annotation), but they lead to a significant improvement in term of recall (70%) and speed (a single mapping may need up to 5 minutes by a human), maintaining an acceptable precision (80%). The system has been used to map 45,978 infobox attributes in 14 different languages for which mappings were not yet available; the resource is made available in an open format.

There remains room for further improvements. For example, the similarity function can be refined with a smarter normalization and a better recognition of typed entities (like temporal expressions, units, and common abbreviations).

We will also evaluate to what extent (precision/recall) DBpedia class mappings can be generated from the property mappings automatically found using our system.

Finally, we will adapt the proposed approach to detect errors in the DBpedia mappings (during our tests we encountered a relevant number of wrong mappings in DBpedia), or to maintain the mappings up-to-date whenever the corresponding Wikipedia templates are updated by the Wikipedia editors.

References

1. Adar, E., Skinner, M., Weld, D.S.: Information arbitrage across multi-lingual Wikipedia. In: Proceedings of the Second ACM International Conference on Web Search and Data Mining, WSDM 2009, pp. 94–103. ACM, New York (2009), http://doi.acm.org/10.1145/1498759.1498813
2. Bizer, C., Lehmann, J., Kobilarov, G., Auer, S., Becker, C., Cyganiak, R., Hellmann, S.: DBpedia - a crystallization point for the web of data. Web Semant. 7(3), 154–165 (2009), http://dx.doi.org/10.1016/j.websem.2009.07.002
3. Bollacker, K., Evans, C., Paritosh, P., Sturge, T., Taylor, J.: Freebase: a collaboratively created graph database for structuring human knowledge. In: Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data, SIGMOD 2008, pp. 1247–1250. ACM, New York (2008), http://doi.acm.org/10.1145/1376616.1376746
4. Bouma, G., Duarte, S., Islam, Z.: Cross-lingual alignment and completion of Wikipedia templates. In: Proceedings of the Third International Workshop on Cross Lingual Information Access: Addressing the Information Need of Multilingual Societies, CLIAWS3 2009, pp. 21–29. Association for Computational Linguistics, Stroudsburg (2009), http://dl.acm.org/citation.cfm?id=1572433.1572437
5. Cancedda, N., Gaussier, E., Goutte, C., Renders, J.M.: Word sequence kernels. J. Mach. Learn. Res. 3, 1059–1082 (2003), http://dl.acm.org/citation.cfm?id=944919.944963
6. Fleiss, J.L.: Measuring Nominal Scale Agreement Among Many Raters. Psychological Bulletin 76(5), 378–382 (1971), http://dx.doi.org/10.1037/h0031619

7. Kontokostas, D., Bratsas, C., Auer, S., Hellmann, S., Antoniou, I., Metakides, G.: Internationalization of Linked Data: The case of the Greek DBpedia edition. Web Semantics: Science, Services and Agents on the World Wide Web 15, 51–61 (2012), http://www.sciencedirect.com/science/article/pii/S1570826812000030
8. Lodhi, H., Shawe-Taylor, J., Cristianini, N.: Text classification using string kernels. Journal of Machine Learning Research 2, 563–569 (2002)
9. Nguyen, T., Moreira, V., Nguyen, H., Nguyen, H., Freire, J.: Multilingual schema matching for Wikipedia infoboxes. Proc. VLDB Endow. 5(2), 133–144 (2011), http://dl.acm.org/citation.cfm?id=2078324.2078329
10. Palmero Aprosio, A., Giuliano, C., Lavelli, A.: Automatic expansion of DBpedia exploiting Wikipedia cross-language information. In: Cimiano, P., Corcho, O., Presutti, V., Hollink, L., Rudolph, S. (eds.) ESWC 2013. LNCS, vol. 7882, pp. 397–411. Springer, Heidelberg (2013)
11. Palmero Aprosio, A., Giuliano, C., Lavelli, A.: Automatic Mapping of Wikipedia Templates for Fast Deployment of Localised DBpedia Datasets. In: Proceedings of the 13th International Conference on Knowledge Management and Knowledge Technologies (2013)
12. Rahm, E., Bernstein, P.A.: A survey of approaches to automatic schema matching. The VLDB Journal 10(4), 334–350 (2001), http://dx.doi.org/10.1007/s007780100057
13. Rinser, D., Lange, D., Naumann, F.: Cross-lingual entity matching and infobox alignment in Wikipedia. Information Systems 38(6), 887–907 (2013), http://www.sciencedirect.com/science/article/pii/S0306437912001299
14. Saunders, C., Tschach, H., Taylor, J.S.: Syllables and other String Kernel Extensions. In: Proc. 19th International Conference on Machine Learning (ICML 2002), pp. 530–537 (2002)
15. Shvaiko, P., Euzenat, J.: A survey of schema-based matching approaches. Journal on Data Semantics 4, 146–171 (2005)
16. Suchanek, F.M., Kasneci, G., Weikum, G.: Yago: a core of semantic knowledge. In: Proceedings of the 16th International Conference on World Wide Web, WWW 2007, pp. 697–706. ACM, New York (2007), http://doi.acm.org/10.1145/1242572.1242667
17. Sultana, A., Hasan, Q.M., Biswas, A.K., Das, S., Rahman, H., Ding, C., Li, C.: Infobox suggestion for Wikipedia entities. In: Proceedings of the 21st ACM International Conference on Information and Knowledge Management, CIKM 2012, pp. 2307–2310. ACM, New York (2012), http://doi.acm.org/10.1145/2396761.2398627
18. Vrandečić, D.: Wikidata: a new platform for collaborative data collection. In: Proceedings of the 21st International Conference Companion on World Wide Web, WWW 2012 Companion, pp. 1063–1064. ACM, New York (2012), http://doi.acm.org/10.1145/2187980.2188242

Type Inference on Noisy RDF Data

Heiko Paulheim and Christian Bizer

University of Mannheim, Germany
Research Group Data and Web Science
{heiko,chris}@informatik.uni-mannheim.de

Abstract. Type information is very valuable in knowledge bases. However, most large open knowledge bases are incomplete with respect to type information, and, at the same time, contain noisy and incorrect data. That makes classic type inference by reasoning difficult. In this paper, we propose the heuristic link-based type inference mechanism *SDType*, which can handle noisy and incorrect data. Instead of leveraging T-box information from the schema, *SDType* takes the actual use of a schema into account and thus is also robust to misused schema elements.

Keywords: Type Inference, Noisy Data, Link-based Classification.

1 Introduction

Type information plays an important role in knowledge bases. Axioms stating that an instance is of a certain type are one of the atomic building blocks of knowledge bases, stating, e.g., that *Thomas Glavinic* is an instance of type *Writer*. Many useful queries to a knowledge base use type information, e.g., *Find all* ***writers*** *from Vienna*, *Is Night Work a* ***novel*** *or a* ***short story****?*, etc.

In many knowledge bases, type information is incomplete for different reasons. For instance, in a crowd-sourced knowledge base, the problem may be simply that no one may have entered the type(s) for a certain instance. When using open world semantics, as in many semantic web knowledge bases, this is not a problem from a logic point of view – however, it drastically limits the usefulness of the knowledge base.

Cross-domain knowledge bases, unlike closed-domain knowledge bases, most often contain a large variety of types. Since it is often not feasible to manually assign types to all instances in a large knowledge base, automatic support creating type information is desirable. Furthermore, since open, crowd-sourced knowledge bases often contain noisy data, logic-based reasoning approaches are likely to multiply errors.

In this paper, we show how type information can be generated heuristically by exploiting other axioms in a knowledge base, in particular links between instances. Unlike classic reasoning approaches, we use a weighted voting approach taking many links into account, which avoids the propagation of errors from single wrong axioms.

H. Alani et al. (Eds.): ISWC 2013, Part I, LNCS 8218, pp. 510–525, 2013.

The rest of this paper is structured as follows. Section 2 motivates our work by showing typical problems of reasoning on real-world datasets. Section 3 introduces the *SDType* approach, which is evaluated in Sect. 4 in different experimental settings. In Sect. 5, we show how *SDType* can be applied to solve a real-world problem, i.e., the completion of missing type information in DBpedia. We conclude our paper with a review of related work in Sect. 6, and a summary and an outlook on future work.

2 Problems with Type Inference on Real-World Datasets

A standard way to infer type information in the Semantic Web is the use of reasoning, e.g., standard RDFS reasoning via entailment rules [20]. To illustrate the problems that can occur with that approach, we have conducted an experiment with *DBpedia* knowledge base [2]. We have used the following subset of entailment rules:

- `?x a ?t1. ?t1 rdfs:subClassOf ?t2` entails `?x a ?t2`
- `?x ?r ?y . ?r rdfs:domain ?t` entails `?x a ?t`
- `?y ?r ?x . ?r rdfs:range ?t` entails `?x a ?t`

We have applied these three rules to the instance `dbpedia:Germany`. These rules in total induce 23 types for `dbpedia:Germany`, only three of which are correct. The list of inferred types contains, among others, the types *award*, *city*, *sports team*, *mountain*, *stadium*, *record label*, *person*, and *military conflict*.

A reasoner requires only one false statement to come to a wrong conclusion. In the example of `dbpedia:Germany`, at most 20 wrong statements are enough to make a reasoner infer 20 wrong types. However, there are more than 38,000 statements about `dbpedia:Germany`, i.e., an error rate of only 0.0005 is enough to end up with such a completely nonsensical reasoning result. In other words: even with a knowledge base that is 99.9% correct, an RDFS reasoner will not provide meaningful results. However, a correctness of 99.9% is difficult, if not impossible, to achieve with real-world datasets populated either (semi-)automatically, e.g., by information extraction from documents, or by the crowd.

In the example above, the class `Mountain` in the above is induced from a single wrong statement among the 38,000 statements about `dbpedia:Germany`, which is `dbpedia:Mže dbpedia-owl:sourceMountain dbpedia:Germany`. Likewise, the class `MilitaryConflict` is induced from a single wrong statement, i.e., `dbpedia:XII_Corps_ (United_Kingdom) dbpedia-owl:battle dbpedia:Germany`.

These problems exist because traditional reasoning is only useful if a) both the knowledge base and the schema do not contain any errors and b) the schema is only used in ways foreseen by its creator [4]. Both assumptions are not realistic for large and open knowledge bases. This shows that, although reasoning seems the straight forward approach to tackle the problem of completing missing types, it is – at least in its standard form – not applicable for large, open knowledge bases, since they are unlikely to have correct enough data for reasoning to

Table 1. Type distribution of the property `dbpedia-owl:location` in DBpedia

Type	Subject (%)	Object (%)
`owl:Thing`	100.0	88.6
`dbpedia-owl:Place`	69.8	87.6
`dbpedia-owl:PopulatedPlace`	0.0	84.7
`dbpedia-owl:ArchitecturalStructure`	50.7	0.0
`dbpedia-owl:Settlement`	0.0	50.6
`dbpedia-owl:Building`	34.0	0.0
`dbpedia-owl:Organization`	29.1	0.0
`dbpedia-owl:City`	0.0	24.2
...	...	...

produce meaningful results. What is required is an approach for inducing types which is tolerant with respect to erroneous and noisy data.

3 Approach

An RDF knowledge base consists of an A-box, i.e., the definition of instances and the relations that hold between them, and a T-box, i.e., a schema or ontology. The *SDType* approach proposed in this paper exploits links between instances to infer their types using weighted voting. Assuming that certain relations occur only with particular types, we can heuristically assume that an instance should have certain types if it is connected to other instances through certain relations. For example, from a statement like `:x dbpedia-owl:location :y`, we may conclude with a certain confidence that `:y` is a place.

3.1 Link-Based Type Inference

SDType uses links between resources as indicators for types, i.e., we propose a *link-based object classification* approach [6]. The basic idea is to use each link from and to a instance as an indicator for the resource's type. For each link, we use the statistical distribution (hence the name ***SD****Type*) of types in the subject and object position of the property for predicting the instance's types.

For each property in a dataset, there is a characteristic distribution of types for both the subject and the object. For example, the property `dbpedia-owl:location` is used in 247,601 triples in DBpedia. Table 1 shows an excerpt of the distribution for that property.[1]

Based on that example distribution, we can assign types with probabilities to `:x` and `:y` when observing a triple like `:x dbpedia-owl:location :y`. Given the distribution in table 1, we could assign $P($`?x a dbpedia-owl:Place`$) = 0.698$, $P($`?y a dbpedia-owl:Place`$) = 0.876$, etc.

More formally, the basic building blocks of *SDType* are conditional properties measuring how likely a type T is, given a resource with a certain property p, expressed as $P\left(T(r)|(\exists p.\top)(r)\right)$, where p may be an incoming or an outgoing

[1] All DBpedia examples in this paper use version 3.8.

property. Furthermore, each property is assigned a certain weight w_p, which reflects its capability of predicting the type (see below). With those elements, we can compute the confidence for a resource r having a type t as

$$\text{conf}\,(T(r)) := \frac{1}{N} \cdot \sum_{\text{all properties } p \text{ of } r} P\,(T(r)|(\exists p.\top)(r)), \tag{1}$$

where N is the number of properties that connects a resource to another one. By using the average probabilities of each type, we address the problem of faulty links, since they do not contribute too much to the overall probability.

In the example with `dbpedia:Germany` used above, the class `Mountain` was inferred due to one wrong statement out of 38,000. With the above definition, that relation would only be weighted with $\frac{1}{38{,}000}$, thus, the type `Mountain` would receive a comparably small overall confidence.

By looking at the *actual* distribution of types co-occurring with a property, instead of the *defined* domains and ranges, properties which are "abused", i.e., used differently than conceived by the schema creator, do not cause any problems for *SDType*. As long as a property is used more or less consistently throughout the knowledge base, the inferences will always be consistent as well. Single inconsistent usages, just like single wrong statements, do not contribute too much to the overall result. Furthermore, when looking at the actual usage of a schema, the results can be more fine-grained than when using the schema only. For example, on the *MusicBrainz* dataset[2], `foaf:name` is always used as a property of `mo:MusicArtist`. While RDFS entailment rules could not infer any specific type from the `foaf:name` property, since it has no explicit domain defined.[3]

While using the actual distribution instead of defined domains and ranges eliminates those problems, it can induce new ones when a dataset is heavily skewed, i.e., the extensions of some classes are several orders of magnitude larger than others. This is a problem in particular with general purpose properties, such as `rdfs:label` or `owl:sameAs`, which are rather equally distributed in the overall knowledge base. If that knowledge base is heavily skewed (e.g., a database about cities and countries which contains 10,000 cities per country on average), and it contains many of such general purpose properties, there is a danger of overrating the more frequent types. Thus, we define a weight w_p for each property (note that p and p^{-1} are treated independently and are each assigned an individual weight), which measures the deviation of that property from the apriori distribution of all types:

$$w_p := \sum_{\text{all types t}} \left(P(t) - P\,(t|\exists p.\top)\right)^2 \tag{2}$$

With those types, we can refine the above definition to

$$\text{conf}\,(T(r)) := \nu \cdot \sum_{\text{all properties } p \text{ of } r} w_p \cdot P\,(T(r)|(\exists p.\top)(r)), \tag{3}$$

with the normalization factor ν defined as

[2] `http://dbtune.org/musicbrainz/`

[3] The defined domain of `foaf:name` is `owl:Thing`, see `http://xmlns.com/foaf/spec/`

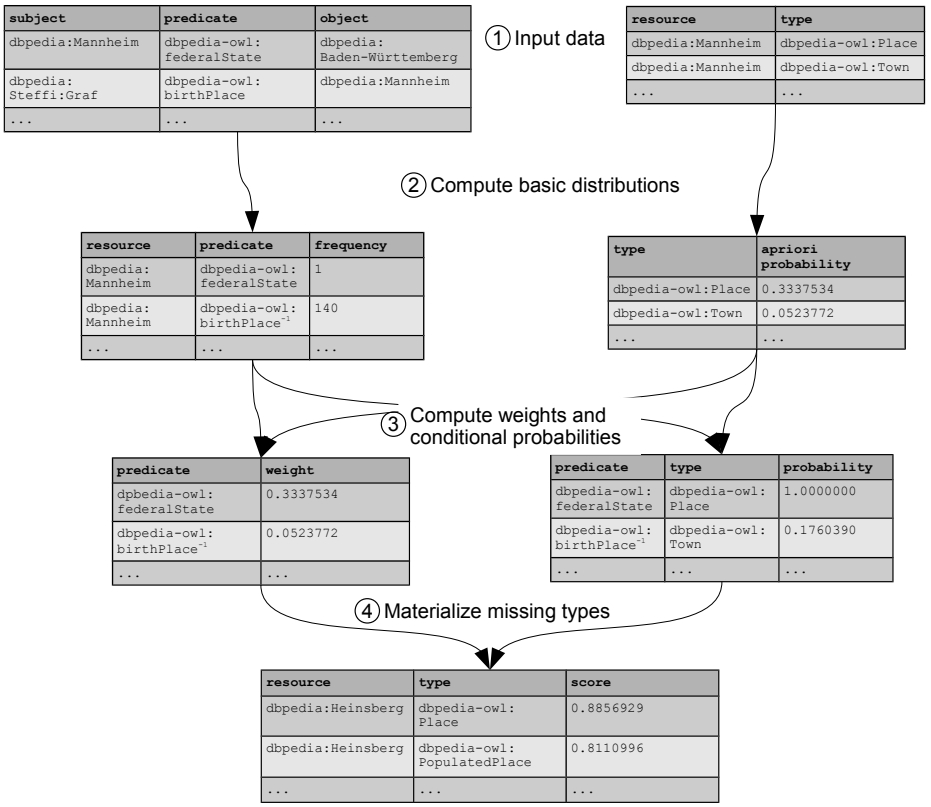


Fig. 1. Implementation of the type completion prototype as a sequence of table creation operations

$$\nu = \frac{1}{\sum_{\text{all properties } p \text{ of } r} w_p} \tag{4}$$

Intuitively, *SDType* implements a weighted voting approach, where for each link, a vote consisting of a distribution of types is cast. The weights reflect the discriminate power of the individual links' properties.

Looking at these weights in DBpedia, for example, we can observe that the maximum weight is given to properties that only appear with one type, such as `dbpedia-owl:maximumBoatLength`, which is only used for `dbpedia-owl:Canal`. On the other end of the spectrum, there are properties such as `foaf:name`, which, in DBpedia, is used for persons, companies, cities, events, etc.

Consider, for example, the triples `:x dbpedia-owl:location :y . :x foaf: name "X"`, and an apriori probability of `dbpedia-owl:Person` and `dbpedia-owl: Place` of 0.21 and 0.16, respectively. With those numbers and distributions such as in table 1, definition (1) would yield a confidence score for

`:x a dbpedia-owl: Person` and `:x a dbpedia-owl:Place` of 0.14 and 0.60, respectively.[4]

When using weights, the numbers are different. In our example from DBpedia, the obtained weights for `dbpedia-owl:location` and `foaf:name` are 0.77 and 0.17, hence, the overall confidence scores for `:x a dbpedia-owl:Person` and `:x a dbpedia-owl:Place` in that example, using definition (3), are 0.05 and 0.78, respectively. This shows that the weights help reducing the influence of general purpose properties and thus assigning more sensible scores to the types that are found by *SDType*, and in the end help reducing wrong results coming from skewed datasets.

In summary, we are capable of computing a score for each pair of a resource and a type. Given a reasonable cutoff threshold, we can thus infer missing types at arbitrary levels of quality – thresholds between 0.4 and 0.6 typically yield statements at a precision between 0.95 and 0.99.

3.2 Implementation

SDType has been implemented based on a relational database, as shown in Fig. 1. The input data consists of two tables, one containing all direct property assertions between instances, the other containing all direct type assertions.

From these input files, basic statistics and aggregations are computed: the number of each type of relation for all resources, and the the apriori probability of all types, i.e., the percentage of instances that are of that type. Each of those tables can be computed with one pass over the input tables or their join.

The basic statistic tables serve as intermediate results for computing the weights and conditional probabilities used in the formulas above. Once again, those weights and conditional probabilities can be computed with one pass over the intermediate tables or their joins.

In a final step, new types can be materialized including the confidence scores. This can be done for all instances, or implemented as a service, which types an instance on demand. Since of each of the steps requires one pass over the database, the overall complexity is linear in the number of statements in the knowledge base.

4 Evaluation

To evaluate the validity of our approach, we use the existing type information in two large datasets, i.e., DBpedia [2] and OpenCyc [9], as a gold standard,[5] and let *SDType* reproduce that information, allowing us to evaluate recall, precision, and F-measure.

[4] The actual numbers for DBpedia are: $P(Person|foaf\#name) = 0.273941$, $P(Place|foaf\#name) = 0.314562$, $P(Person|dbpedia\#location) = 0.000236836$, $P(Place|dbpedia\#location) = 0.876949$.

[5] In the case of DBpedia, the dataset is rather a silver standard. However, it provides the possibility of a larger-scale evaluation. A finer-grained evaluation with manual validation of the results by an expert can be found in Sect. 5.

Table 2. Characteristics of the datasets used for evaluation

	DBpedia	OpenCyc
Number of instances	3,600,638	193,049
Number of distinct classes	359	119,941
Number of distinct properties	1775	18,526
Average depth of leaf classes in the class hierarchy	2.4	10.7
Average number of type statements per (typed) instance	5.6	59.9
Average number of instances per type	38,003.3	755.2
Average number of ingoing properties per instance	8.5	4.8
Average number of outgoing properties per instance	8.8	4.0

4.1 Datasets

DBpedia is generated automatically from Wikipedia infoboxes, and has a large coverage, at the price of reduced precision, e.g., due to parsing errors or mistakes in Wikipedia itself. OpenCyc, on the other hand, is more focused on precise data allowing for exact reasoning, but has a lower coverage than DBpedia. The DBpedia dataset contains all types from the infobox types dataset (i.e., DBpedia ontology, schema.org, and UMBEL).[6]

While DBpedia has all type information for the DBpedia ontology fully materialized w.r.t. `rdfs:subClassOf`, we manually materialized all direct types in OpenCyc, using simple RDFS-like inference for subclasses and subproperties (the latter are not used at all in the DBpedia ontology). Table 2 lists some relevant characteristics of the datasets.

It can be observed that the class hierarchy of OpenCyc is several orders of magnitude larger and more fine-grained than the class hierarchy of DBpedia. At the same time, the average number of instances in each class is much smaller for OpenCyc. Since the average number of properties per instance is also lower, the problem of inferring types with *SDType* on OpenCyc is harder for two reasons: there is les evidences for each instance, and the number of classes to predict is higher.

For both datasets, we have used random samples of 10,000 instances. Furthermore, we restrict our approach to using only *ingoing* properties. The reason is that classification based on outgoing properties would oversimplify the problem. In DBpedia, outgoing properties and types are generated in the same step, so the correct type can be trivially predicted from outgoing properties. The same holds for OpenCyc, which uses per class templates for populating instance data [17]. Furthermore, when trying to infer *missing* types, the instances with missing types most often have no outgoing properties.

4.2 Results

Figure 2 shows the results of *SDType* on DBpedia.[7] While it can be observed that *SDType* works sufficiently well on the overall dataset (i.e., instances that have at

[6] `http://dbpedia.org/Downloads38`

[7] The predicted types include those defined in the DBpedia ontology, schema.org, and UMBEL, as well as `owl:Thing`.

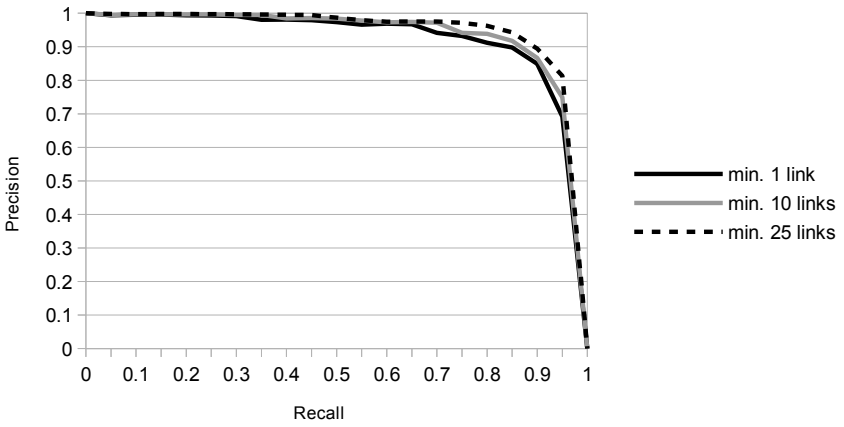


Fig. 2. Precision/recall curves of *SDType* on DBpedia, for instances with at least one, at least 10, and at least 25 incoming links

least one ingoing link), achieving an F-measure of 88.5%, the results are slightly better on instances that have at least 10 or 25 ingoing links, with an F-measure of 88.9% and 89.9%, respectively. The differences show more significantly in the precision@95% (i.e. the precision that can be achieved at 95% recall), which is 0.69 (minimum one link), 0.75 (minimum ten links), and 0.82 (minimum 25 links), respectively.

Figure 3 depicts the corresponding results for OpenCyc. The first observation is that the overall results are not as good as on DBpedia, achieving a maximum F-measure of 60.1% (60.3% and 60.4% when restricting to instances that have at least 10 or 25 ingoing links). The second observation is that the results for instances with different numbers of ingoing properties do not differ much – in fact, most of the differences are too small to be visible in the figure. While 95% recall cannot be reached on OpenCyc with *SDType*, the precision@90% is 0.18 (minimum one link), 0.23 (minimum ten and 25 links), respectively.

The strong divergence of the results between DBpedia and OpenCyc, as discussed above, was to be expected, since OpenCyc has on the one hand more (and more specific) types per instance, on the other hand less evidence per instance, since the number of properties connecting instances is smaller.

As the diagrams show, looking at instances with more links improves the results on DBpedia, but not on OpenCyc (apart from a small improvement in precision at a recall of around 0.9). The reason for that is that DBpedia, with its stronger focus on coverage than on correctness, contains more faulty statements. When more links are present, the influence of each individual statement is reduced, which allows for correcting errors. OpenCyc, on the other hand, with its stronger focus on precision, benefits less from that error correction mechanism.

Since we assume that it is more difficult to predict more specific types (such as *Heavy Metal Band*) than predicting more general ones (like *Band* or even *Organization*), we have additionally examined the best F-measure that can be

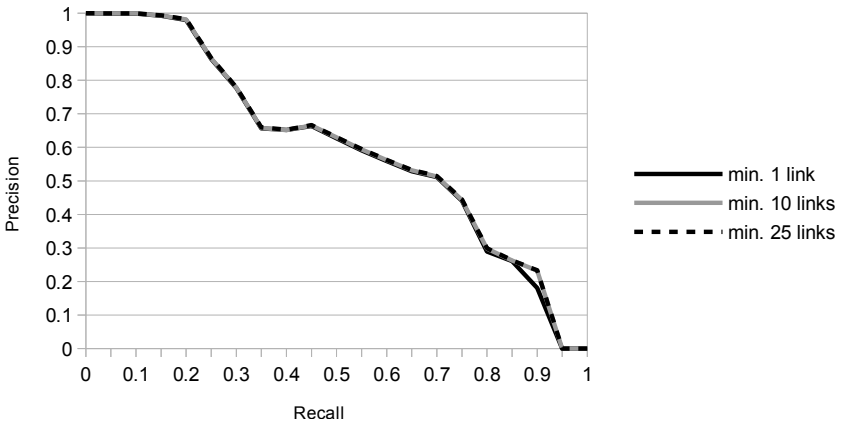


Fig. 3. Precision/recall curves of *SDType* on OpenCyc, taking into account only incoming, only outgoing, and both incoming and outgoing properties

achieved when restricting the approach to a certain maximum class hierarchy depth. The results are depicted in Fig. 4. It can be observed that *SDType* in fact works better on more general types (achieving an F-measure of up to 97.0% on DBpedia and 71.6% on OpenCyc when restricting the approach to predicting only top-level classes). However, the effects are weaker than we expected.

5 Application: Completing Missing Types in DBpedia

In the following, we apply *SDType* to infer missing type information in DBpedia. While DBpedia has a quite large coverage, there are millions of missing type statements. To infer those missing types, we have combined the approach sketched above with a preclassification step separating typeable from untypeable resources in order to reduce false inferences.

5.1 Estimating Type Completeness in DBpedia

Aside from the type information in DBpedia using the DBpedia ontology, which is generated using Wikipedia infoboxes, resources in DBpedia are also mapped to the YAGO ontology [18]. Those mappings are generated from Wikipedia page categories. Thus, they are complementary to DBpedia types – an article may have a correct infobox, but missing category information, or vice versa. Both methods of generating type information are prone to (different types of) errors. However, looking at the overlaps and differences of type statements created by both methods may provide some approximate estimates about the completeness of DBpedia types.

To estimate the completeness of type information in DBpedia, we used a partial mapping between the YAGO ontology [18] and the DBpedia ontology.[8]

[8] `http://www.netestate.de/De/Loesungen/DBpedia-YAGO-Ontology-Matching`

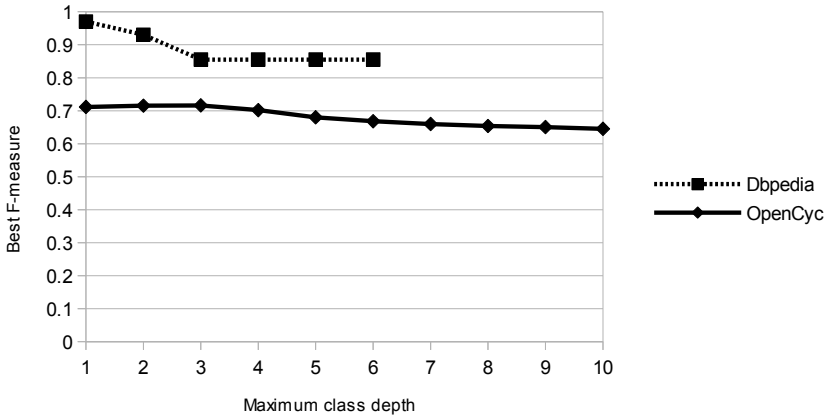


Fig. 4. Maximum achievable F-measure by maximum class depth for DBpedia and OpenCyc. The graph depicts the maximum F-measure that can be achieved when restricting the approach to finding classes of a maximum hiearchy depth of 1, 2, etc.

Assuming that the YAGO types are at least more or less correct, we can estimate the completeness of a DBpedia type `dbpedia#t` using the mapped YAGO type `yago#t` by looking at the relation of all instances of `dbpedia#t` and all instances that have at least one of the types `dbpedia#t` and `yago#t`:

$$completeness(\texttt{dbpedia\#t}) \leq \frac{|\texttt{dbpedia\#t}|}{|\texttt{dbpedia\#t} \cup \texttt{yago\#t}|} \tag{5}$$

The denominator denotes an estimate of all instances that *should* have the type `dbpedia#t`. Since the actual number of resources that should have that type can be larger than that (i.e., neither the DBpedia nor the YAGO type is set), the completeness can be smaller than the fraction, hence the inequation.

Calculating the sum across all types, we observe that DBpedia types are at most 63.7% complete, with at least 2.7 million missing type statements (while YAGO types, which can be assessed accordingly, are at most 53.3% complete). The classes the most missing type statements are shown in Fig. 5

Classes that are very incomplete include

- `dbpedia-owl:Actor` (completeness $\leq$ 4%), with 57,000 instances missing the type, including, e.g., Brad Pitt and Tom Hanks
- `dbpedia-owl:Game` (completeness $\leq$ 7%), with 17,000 instances missing the type, including Tetris and Sim City
- `dbpedia-owl:Sports` (completeness $\leq$ 5.3%), with 3,300 instances missing the type, including Beach Volleyball and Biathlon

A similar experiment using the classes `dbpedia-owl:Person` and `foaf:Person` (assuming that each person should have both types) yielded that the class `dbpedia-owl:Person` is at most 40% complete. These examples show that the problem of missing types in DBpedia is large, and that it does not only affect

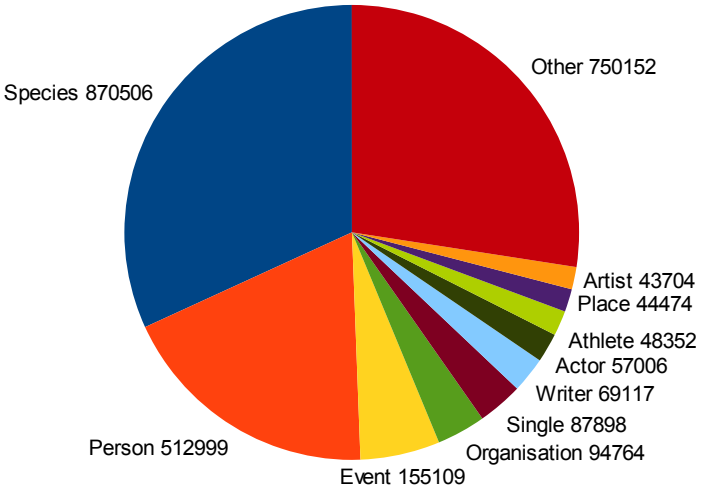


Fig. 5. Largest number of (estimated) missing type statements per class

marginally important instances. In DBpedia, common reasons for missing type statements are

- Missing infoboxes – an article without an infobox is not assigned any type.
- Too general infoboxes – if an article about an actor uses a person infobox instead of the more specific actor infobox, the instance is assigned the type `dbpedia-owl:Person`, but not `dbpedia-owl:Actor`.
- Wrong infobox mappings – e.g., the videogame infobox is mapped to `dbpedia- owl:VideoGame`, not `dbpedia-owl:Game`, and `dbpedia-owl: VideoGame` is not a subclass of `dbpedia-owl:Game` in the DBpedia ontology.
- Unclear semantics – some DBpedia ontology classes do not have clear semantics. For example, there is a class `dbpedia-owl:College`, but it is not clear which notion of *college* is denoted by that class. The term *college*, according to different usages, e.g., in British and US English, can denote private secondary schools, universities, or institutions within universities.[9]

5.2 Typing Untyped Instances in DBpedia

In our second experiment, we have analyzed how well *SDType* is suitable for adding type information to untyped resources. As discussed above, resources may be missing a type because they use no infobox, an infobox not mapped to a type, or are derived from a Wikipedia red link. In particular in the latter case, the only usable information are the incoming properties.

Simply typing all untyped resources with *SDType* would lead to many errors, since there are quite a few resources that should not have a type, as discussed

[9] See `http://oxforddictionaries.com/definition/english/college`

in [1]. Examples are resources derived from list pages,[10] pages about a category rather than an individual,[11] or general articles.[12]

In order to address that problem, we have manually labeled 500 untyped resources into typeable and non-typeable resources. For those resources, we have created features using the *FeGeLOD* framework [13], and learned a ruleset for classifying typeable and non-typeable resources using the *Ripper* rule learner [3]. The resulting rule set has accuracy of 91.8% (evaluated using 10-fold cross validation).

From all 550,048 untyped resources in DBpedia, this classifier identifies 519,900 (94.5%) as typeable. We have generated types for those resources and evaluated them manually on a sample of 100 random resources. The results for various thresholds are depicted in Fig. 6. It can be observed that 3.1 types per instance can be generated with a precision of 0.99 at a threshold of 0.6, 4.0 types with a precision of 0.97 at a threshold of 0.5, and 4.8 types with a precision of 0.95 at a threshold of 0.4.[13]. In contrast, RDFS reasoning on the test dataset generates 3.0 types per instance with a precision of 0.96, which shows that *SDType* is better in both precision and productivity.

With those thresholds, we can generate a total of 2,426,552 and 1,682,704 type statements, respectively, as depicted in Table 3. It can be observed that with the higher threshold guaranteeing higher precision, more general types are generated, while more specific types such as *Athlete* or *Artist*, are rarely found. In most cases, the generated types are consistent, i.e., an *Artist* is also a *Person*, while contradicting predictions (e.g., *Organization* and *Person* for the same instance) are rather rare.

6 Related Work

The problems of inference on noisy data in the Semantic Web has been identified, e.g., in [16] and [8]. While general-purpose reasoning on noisy data is still actively researched, there have been solutions proposed for the specific problem of type inference in (general or particular) RDF datasets in the recent past, using strategies such as machine learning, statistical methods, and exploitation of external knowledge such as links to other data sources or textual information.

[11] use a similar approach as ours, but on a different problem: they try to predict possible predicates for resources based on co-occurrence of properties. They report an F-measure of 0.85 at linear runtime complexity.

Many ontology learning algorithms are capable of dealing with noisy data [19]. However, when using the learned ontologies for inferring missing information using a reasoner, the same problems as with manually created ontologies occur.

[10] e.g., `http://dbpedia.org/resource/Lists_of_writers`

[11] e.g., `http://dbpedia.org/resource/Writer`

[12] e.g., `http://dbpedia.org/resource/History_of_writing`

[13] A web service for DBpedia type completion, as well as the code used to produce the additional types, is available at `http://wifo5-21.informatik.uni-mannheim.de:8080/DBpediaTypeCompletionService/`

Table 3. Results for typing untyped resources, including main types found. The table lists all types which were predicted for at least 1% of the instances in the test set.

Threshold	0.4	0.6
Estimated precision	≥ 0.95	≥ 0.99
Total typed instances	440,849	373,366
Total type statements	2,426,552	1,682,704
Average types per typed instance	5.5	4.5
Distinct types assigned	144	121
Main types:		
Person	236,608 (53.7%)	173,944 (46.6%)
– Athlete	71,226 (16.2%)	544 (<0.1%)
– Artist	21,219 (4.8%)	22 (<0.1%)
– Musical Artist	10,533 (2.4%)	21 (<0.1%)
– Writer	4,973 (1.1%)	0 (0.0%)
Place	79,115 (17.9%)	72,593 (19.4%)
– Settlement	52,622 (11.9%)	23,060 (6.2%)
– Natural Place	4,846 (1.1%)	2,293 (1.0%)
Organization	73,148 (16.6%)	46,988 (12.6%)
– Company	25,077 (5.7%)	21,509 (5.8%)
– Sports Team	15,176 (3.4%)	14,635 (3.9%)
– Record Label	13,444 (3.0%)	13,158 (3.5%)
– Band	12,770 (2.9%)	6 (<0.1%)
Creative Work	15,542 (3.5%)	13,130 (3.4%)
– Album	12,516 (2.8%)	191 (<0.1%)
Species	8,249 (1.8%)	7,988 (2.1%)
– Animal	7,815 (1.7%)	6,744 (1.8%)

One of the first approaches to type classification in relational data is discussed in [10]. The authors train a machine learning model on instances that already have a type, and apply it to the untyped instances in an iterative manner. The authors report an accuracy of 0.81, treating type completion as a single-class problem (i.e., each instance is assigned exactly one type).

The work discussed in [12] assumes that for many instances, there are *some*, but not *all* types. Association rule mining is employed to find common patterns of the type *if type A and B are set, type C is also set*, and apply them to the knowledge base. The authors report that they can add around 3 additional types to an average instance at a precision of 85.6%.

In [1], an approach is introduced which first exploits cross-language links between DBpedia in different languages to increase coverage, e.g., if an instance has a type in one language version and does not have one in another language version. Then, they use nearest neighbor classification based on different features, such as templates, categories, and bag of words of the corresponding Wikipedia article. On existing type information, the authors report a recall of 0.48, a precision of 0.91, and an F-measure of 0.63.

The *Tipalo* system [5] leverages the natural language descriptions of DBpedia entities to infer types, exploiting the fact that most abstracts in Wikipedia follow

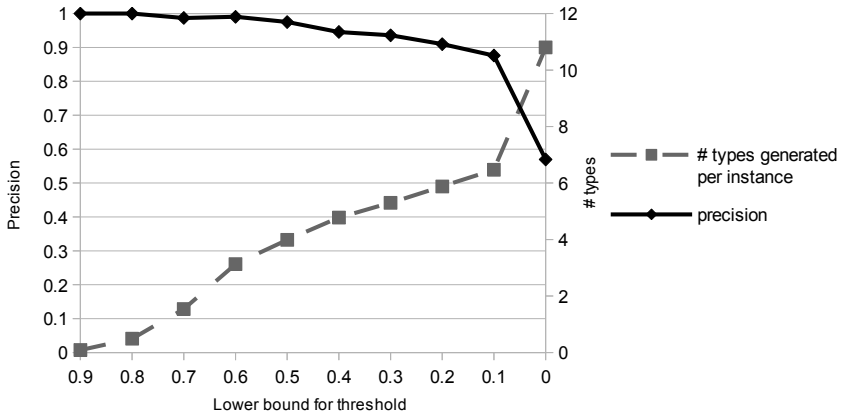


Fig. 6. Precision and average number of type statements per resource generated on untyped resources in DBpedia

similar patterns. Those descriptions are parsed and mapped to the WordNet and DOLCE ontologies in order to find appropriate types. The authors report an overall recall of 0.74, a precision of 0.76, and an F-measure of 0.75.

The authors of [7] exploit types of resources derived from linked resources, where links between Wikipedia pages are used to find linked resources (which are potentially more than resources actually linked in DBpedia). For each resource, they use the classes of related resources as features, and use k nearest neighbors for predicting types based on those features. The authors report a recall of 0.86, a precsion of 0.52, and hence an F-measure of 0.65.

The approach discussed in [15] addresses a slightly different problem, i.e., the mapping DBpedia entities to the category system of OpenCyc. They use different indicators – infoboxes, textual descriptions, Wikipedia categories and instance-level links to OpenCyc – and apply an a posteriori consistency check using Cyc's own consistency checking mechanism. The authors report a recall of 0.78, a precision of 0.93, and hence an F-measure of 0.85.

The approaches discussed above, except for [12], are using specific features for DBpedia. In contrast, *SDType* is agnostic to the dataset and can be applied to any RDF knowledge base. Furthermore, none of the approaches discussed above reaches the quality level of *SDType* (i.e., an F-measure of 88.5% on the DBpedia dataset).

With respect to DBpedia, it is further noteworthy that *SDType* is also capable of typing resources derived from Wikipedia pages with very sparse information (i.e., no infoboxes, no categories, etc.) – as an extreme case, we are also capable of typing instances derived from Wikipedia red links only by using information from the ingoing links.

7 Conclusion and Outlook

In this paper, we have discussed the *SDType* approach for heuristically completing types in large, cross-domain databases, based on statistical distributions.

Unlike traditional reasoning, our approach is capable of dealing with noisy data as well as faulty schemas or unforeseen usage of schemas.

The evaluation has shown that *SDType* can predict type information with an F-measure of up to 88.9% on DBpedia and 63.7% on OpenCyc, and can be applied to virtually any cross-domain dataset. For DBpedia, we have furthermore enhanced *SDType* to produce valid types only for untyped resources. To that end, we have used a trained preclassifier telling typeable from non-typeable instances at an accuracy of 91.8%, and are able to predict 2.4 million missing type statements at a precision of 0.95, or 1.7 million missing type statements at a precision of 0.99, respectively. We have shown that with these numbers, we outperform traditional RDFS reasoning both in precision and productivity.

The results show that *SDType* is good at predicting higher-level classes (such as *Band*), while predicting more fine-grained classes (such as *Heavy Metal Band*) is much more difficult. One strategy to overcome this limitation would be to use *qualified relations* instead of only relation information, i.e., a combination of the relation and the type of related objects. For example, links from a music group to an instance of *Heavy Metal Album* could indicate that this music group is to be classified as a *Heavy Metal Band*. However, using such features results in a much larger feature space [13] and thus creates new challenges with respect to scalability of *SDType*.

The type statements created by *SDType* are provided in a web service interface, which allows for building applications and services at a user-defined trade-off of recall and precision, as sketched in [14].

The statistical measures used in this paper cannot only be used for predicting missing types. Other options we want to explore in the future include the validation of existing types and links. Like each link can be an indicator for a type that does not exist in the knowledge base, it may also be an indicator that an existing type (or the link itself) is wrong.

In summary, we have shown an approach that is capable of making type inference heuristically on noisy data, which significantly outperforms previous approaches addressing this problems, and which works on large-scale datasets such as DBpedia. The resulting high precision types for DBpedia have been added to the DBpedia 3.9 release and are thus publicly usable via to the DBpedia services.

Acknowledgements. The authors would like to thank Christian Meilicke for his valuable feedback on this paper.

References

1. Palmero Aprosio, A., Giuliano, C., Lavelli, A.: Automatic expansion of dbpedia exploiting wikipedia cross-language information. In: Cimiano, P., Corcho, O., Presutti, V., Hollink, L., Rudolph, S. (eds.) ESWC 2013. LNCS, vol. 7882, pp. 397–411. Springer, Heidelberg (2013)
2. Bizer, C., Lehmann, J., Kobilarov, G., Auer, S., Becker, C., Cyganiak, R., Hellmann, S.: DBpedia - A crystallization point for the Web of Data. Web Semantics 7(3), 154–165 (2009)

3. Cohen, W.W.: Fast effective rule induction. In: 12th International Conference on Machine Learning (1995)
4. Fensel, D., van Harmelen, F.: Unifying Reasoning and Search. IEEE Internet Computing 11(2), 94–95 (2007)
5. Gangemi, A., Nuzzolese, A.G., Presutti, V., Draicchio, F., Musetti, A., Ciancarini, P.: Automatic typing of dbpedia entities. In: Cudré-Mauroux, P., Heflin, J., Sirin, E., Tudorache, T., Euzenat, J., Hauswirth, M., Parreira, J.X., Hendler, J., Schreiber, G., Bernstein, A., Blomqvist, E. (eds.) ISWC 2012, Part I. LNCS, vol. 7649, pp. 65–81. Springer, Heidelberg (2012)
6. Getoor, L., Diehl, C.P.: Link mining: a survey. ACM SIGKDD Explorations Newsletter 7(2), 3–12 (2005)
7. Giovanni, A., Gangemi, A., Presutti, V., Ciancarini, P.: Type inference through the analysis of wikipedia links. In: Linked Data on the Web (LDOW) (2012)
8. Ji, Q., Gao, Z., Huang, Z.: Reasoning with noisy semantic data. In: Antoniou, G., Grobelnik, M., Simperl, E., Parsia, B., Plexousakis, D., De Leenheer, P., Pan, J. (eds.) ESWC 2011, Part II. LNCS, vol. 6644, pp. 497–502. Springer, Heidelberg (2011)
9. Matuszek, C., Cabral, J., Witbrock, M., DeOliveira, J.: An introduction to the syntax and content of cyc. In: Proceedings of the 2006 AAAI Spring Symposium on Formalizing and Compiling Background Knowledge and its Applications to Knowledge Representation and Question Answering (2006)
10. Neville, J., Jensen, D.: Iterative classification in relational data. In: Proc. AAAI-2000 Workshop on Learning Statistical Models from Relational Data, pp. 13–20 (2000)
11. Oren, E., Gerke, S., Decker, S.: Simple algorithms for predicate suggestions using similarity and co-occurrence. In: Franconi, E., Kifer, M., May, W. (eds.) ESWC 2007. LNCS, vol. 4519, pp. 160–174. Springer, Heidelberg (2007)
12. Paulheim, H.: Browsing linked open data with auto complete. In: Semantic Web Challenge (2012)
13. Paulheim, H., Fürnkranz, J.: Unsupervised Feature Generation from Linked Open Data. In: International Conference on Web Intelligence, Mining, and Semantics, WIMS 2012 (2012)
14. Paulheim, H., Pan, J.Z.: Why the semantic web should become more imprecise. In: What will the Semantic Web Look Like 10 Years from Now? (2012)
15. Pohl, A.: Classifying the wikipedia articles in the opencyc taxonomy. In: Web of Linked Entities Workshop (WoLE 2012) (2012)
16. Polleres, A., Hogan, A., Harth, A., Decker, S.: Can we ever catch up with the web? Semantic Web Journal 1(1,2), 45–52 (2010)
17. Shah, P., Schneider, D., Matuszek, C., Kahlert, R.C., Aldag, B., Baxter, D., Cabral, J., Witbrock, M.J., Curtis, J.: Automated population of cyc: Extracting information about named-entities from the web. In: Proceedings of the Nineteenth International Florida Artificial Intelligence Research Society Conference (FLAIRS), pp. 153–158. AAAI Press (2006)
18. Suchanek, F.M., Kasneci, G., Weikum, G.: Yago: a core of semantic knowledge. In: Proceedings of the 16th international conference on World Wide Web, WWW 2007, pp. 697–706. ACM (2007)
19. Völker, J., Niepert, M.: Statistical schema induction. In: Antoniou, G., Grobelnik, M., Simperl, E., Parsia, B., Plexousakis, D., De Leenheer, P., Pan, J. (eds.) ESWC 2011, Part I. LNCS, vol. 6643, pp. 124–138. Springer, Heidelberg (2011)
20. W3C. RDF Semantics (2004), `http://www.w3.org/TR/rdf-mt/`

What's in a 'nym'? Synonyms in Biomedical Ontology Matching

Catia Pesquita[1], Daniel Faria[1], Cosmin Stroe[2], Emanuel Santos[1], Isabel F. Cruz[2], and Francisco M. Couto[1]

[1] Dept. de Informática, Faculdade de Ciências, Universidade de Lisboa, Portugal
[2] Dept. of Computer Science, University of Illinois at Chicago, USA
cpesquita@di.fc.ul.pt

Abstract. To bring the Life Sciences domain closer to a Semantic Web realization it is fundamental to establish meaningful relations between biomedical ontologies. The successful application of ontology matching techniques is strongly tied to an effective exploration of the complex and diverse biomedical terminology contained in biomedical ontologies. In this paper, we present an overview of the lexical components of several biomedical ontologies and investigate how different approaches for their use can impact the performance of ontology matching techniques. We propose novel approaches for exploring the different types of synonyms encoded by the ontologies and for extending them based both on internal synonym derivation and on external ontologies.

We evaluate our approaches using AgreementMaker, a successful ontology matching platform that implements several lexical matchers, and apply them to a set of four benchmark biomedical ontology matching tasks. Our results demonstrate the impact that an adequate consideration of ontology synonyms can have on matching performance, and validate our novel approach for combining internal and external synonym sources as a competitive and in many cases improved solution for biomedical ontology matching.

Keywords: Ontology Matching, Synonym Derivation, Ontology Extension, Biomedical Ontologies.

1 Introduction

Research in the Life Sciences, and in particular in biomedical research, has much to gain from Semantic Web technologies due to the amount and complexity of the data involved. One crucial development has been the creation of ontologies that describe biomedical knowledge and support several applications, both theoretical and practical, such as the representation of encyclopedic knowledge, semantic search and query, data exchange and integration, and reasoning support [1]. However, to fully benefit from the overall knowledge contained in those ontologies, meaningful connections need to be established across the concepts from various ontologies. To establish these relations, we can use ontology matching techniques that are able to find correspondences between semantically related entities belonging to different ontologies [2].

H. Alani et al. (Eds.): ISWC 2013, Part I, LNCS 8218, pp. 526–541, 2013.

The matching of biomedical ontologies poses considerable challenges, given their particular characteristics. The domains they cover are usually complex and large, with many biomedical ontologies possessing tens of thousands of classes dedicated to highly specific areas such as genomics, phenotypes or cellular structures. Moreover, biomedical terminology is characterized by ambiguity and complexity, features that further complicate the application of many ontology engineering techniques. However, the biomedical domain also presents some interesting opportunities such as the exploration of an abundant scientific literature or the availability of many related biomedical ontologies. Despite the efforts of the community to provide orthogonal ontologies [3], many contain overlapping knowledge. For instance, in BioPortal [4], a portal for biomedical ontologies, there are currently 306 ontologies distributed by categories, of which 59 in health, 38 in anatomy and 21 in biological processes.

In recent years the OAEI (Ontology Alignment Evaluation Initiative) [5] has been the major playfield for biomedical ontology alignment, both in its anatomy track, and more recently in the large biomedical ontologies track. An important finding of the OAEI is that many of the anatomy ontologies correspondences are rather trivial and can be found by simple string comparison techniques. To confirm this finding, a simple string matching algorithm, LOOM, was applied to several ontologies available in the NCBO BioPortal, obtaining high levels of precision in most cases [6]. Explanations for this fact include the simple structure of most biomedical ontologies, the high number of synonyms they contain, and their low language variability. Several strategies have been used by the top ranked systems at OAEI to increase recall that go beyond internal lexical similarity, including the use of external knowledge resources (SAMBO [7]) and ontologies (GOMMA [8], AgreementMaker [9]), global similarity computation techniques (AgreementMaker [10], SOBOM [11]), and more complex measures of label and structural similarity (AgreementMaker, LogMap [12]). A combination of these strategies has enabled two of the best systems, GOMMA and AgreementMaker, to reach a F-measure above 90% in the anatomy track. With the introduction of the large biomedical ontologies track in 2012, competing systems developed strategies to handle the very large size of the ontologies therein, including the selection of specific portions of the ontologies to apply matching [13]. Likewise, a new emphasis on the coherence of the generated alignments, prompted several systems to incorporate strategies to improve their alignments coherence [8, 12].

However, this shift in ontology matching systems to ensure the ontological quality of their strategies and results has not translated to the handling of terminological properties, despite the common knowledge of their importance to support matching.

The purpose of this paper is to show the positive impact that is brought by a deep understanding of the terminology contained in ontologies, when used in conjunction with current ontology matching approaches. To support this premise, we have surveyed the terminological component of several biomedical ontologies (including those used by the OAEI tracks) with a special emphasis on synonyms, and tested several novel approaches to improve lexical based matching

approaches. These approaches include: (1) the ranking and weighting of names and synonyms based on their degree of closeness; (2) the derivation of new synonyms based on the ones encoded by a single or both ontologies; and (3) the addition of new synonyms based on cross-references or lexical matches to related external ontologies.

The paper is organized as follows: Section 2 describes the terminological component of several biomedical ontologies and discusses their implications for ontology matching. Section 3 describes our three approaches to improve lexical-based matches. Section 4 describes the evaluation methods, while Section 5 presents and discusses the results obtained using those methods. Finally Section 6 contextualizes our contributions including their limitations and future work.

2 Synonyms in Biomedical Ontologies

Biomedical terminology is complex and ambiguous—frequently the same entity has several names (e.g., gluconeogenesis, glucose synthesis and glucose biosynthesis, all refer to the same metabolic process), a common word refers to a biomedical entity (e.g., hedgehog, and fruitfly are both gene names), or even the same word can be applied to two different entities (e.g., lingula, can either be a structure of the brain or of the lung). These challenges provide one of the major motivations to develop biomedical ontologies, given their explicit definition of concepts through ontological properties.

Biomedical ontologies characteristically have a strong terminological component in the form of names and multiple types of synonyms. Most ontologies define a primary name or label for each class, which is usually encoded as either a localname property or a label property when localnames are reserved for alphanumeric identifiers. Since biomedical entities usually have more than one name, ontologies encode alternative labels as different kinds of synonym properties, which help distinguish between the main label of a class and its alternatives, be they equivalent or merely related. Ontologies under the Open Biomedical Ontologies initiative [3] usually encode the following synonyms types: *hasExactSynonym*, where the alias exhibits true synonymy; *hasBroadSynonym* and *hasNarrowSynonym* where the aliases are broader or narrower than the primary name; and *hasRelatedSynonym*, where the alias is related to the primary class name but not necessarily broader or narrower. Other biomedical ontologies usually also encode distinct types of synonyms, reflecting different degrees of closeness in meaning to the main term. To the set of main labels and synonyms we henceforth call *names*. Some biomedical ontologies have cross-reference properties that connect ontology classes to classes from other ontologies. These links can be used to transfer name properties between cross-referenced classes. Table 1 presents some statistics on synonyms and cross-references for several biomedical ontologies, namely those provided by the OAEI, which will be used as a testbed for our proposed approaches. Most ontologies encode several synonyms for each class, with the notable exception of SNOMED, where synonyms are very rare. At the other end of the spectrum we have UBERON, an ontology designed to

Table 1. Name properties in biomedical ontologies

Ontology	Classes	Name properties		Names per class
NCI_Human (OAEI)	3304	label	3304	1.59
		hasRelatedSynonym	5264	
MA (OAEI)	2739	label	2739	1.13
		hasRelatedSynonym	345	
FMA (OAEI)	79042	label	133629	1.69
NCI (OAEI)	66917	label	175972	2.63
SNOMED (OAEI)	122464	label	122566	1.00
FMA	78977	label	105490	1.92
		hasExactSynonym	45996	
NCI	96717	FULL_SYN*	303121	4.13
		label	96717	
UBERON	8659	label	8659	12.11
		hasExactSynonym	20955	
		hasRelatedSynonym	6150	
		hasNarrowSynonym	562	
		hasBroadSynonym	442	
		hasDbXref**	68068	

*equivalent to hasExactSynonym, **link to an external ontology or resource

integrate cross-species anatomy, which encodes a high number of distinct synonym properties, as well as cross-references to several other ontologies, including MA, NCI, SNOMED and FMA.

Although state of the art ontology matching systems use synonyms in their strategies, they do so without considering the ontological property that encondes them and its meaning. In ontologies encoding more than one kind of name property it makes sense that ontology matching techniques differentiate between them.

3 Methods for Exploring the Use of Synonyms in Ontology Matching

3.1 Synonym Ranking and Weighting

Considering that several ontologies encode distinct types of synonyms, we base our approach on the notion that a synonym should contribute to the similarity score between two ontology classes in proportion to its closeness to the main name of the class it belongs to. To arrive at this weight, we first rank the synonyms encoded in an ontology according to the synonym property they are assigned to. Following the logical definition of commonly used synonym properties, we propose the following default ranking of name properties: (1) localname, (2) label, (3) exact synonym, (4) related synonym, (5) broad synonym, (6) narrow synonym, (7) other synonyms. Whenever an ontology does not possess one of these properties, the rank of the following properties can be increased. This is especially relevant when matching an ontology where the localname corresponds

to a unique alphanumeric identifier to an ontology where the localname is the main label of the class. These ranks can then be used to attribute weights to a classs names given the input of a single interval according to:

$$weight = 1.0 - (interval * (rank - 1)) \quad (1)$$

3.2 Ontology Lexicon Extension through Synonym Derivation

Despite the already high number of synonyms present in most biomedical ontologies, it is a cumbersome task for ontology developers to cover all possible variants. Moreover, when ontologies belong to similar but parallel domains (for instance, when they cover the anatomy of distinct mammal species) they will encode the synonyms that belong to their strict domain, but many times forgo synonyms of broader spectrum. One strategy that can be used to circumvent this omission is to extend the synsets of ontology classes with WordNet synonyms [14]. However, in the biomedical domain this strategy has been shown to slightly increase recall but at a higher cost of precision [15], which is likely due to the highly specialized vocabulary contained in biomedical ontologies and its limited coverage by WordNet.

Our novel approach is based on the notion that we can explore the synonymy relations established between sets of names within the ontologies to derive new synonyms. A preliminary implementation of this approach was integrated in AgreementMaker in 2011 [9]. The main idea behind this approach is that by finding common terms (both single and multi-word) between ontology synonyms we can infer a synonym relation between the remaining distinct terms. These terms can then be used to generate new synonym names. Since this approach is solely based on ontology terminology, we expect it to avoid the issues encountered when using a non-specific resource such as WordNet. For example, in the mouse anatomy ontology the class named as 'stomach serosa' has the synonym 'gastric serosa', which supports the inference that the terms 'stomach' and 'gastric' are synonymous. These synonymous terms are then used to create novel synonyms, by substituting terms with their synonyms in existing names. For instance, we can create a new synonym for the class 'stomach secretion' using the synonyms 'stomach' and 'gastric' to create the new synonym 'gastric secretion'.

We implement our approach in two main steps: (1) the construction of one or two thesauri containing synonym terms; and (2) the derivation of new synonyms based on thesaurus entries. The thesauri can be built based on a single ontology, one for each ontology, or based on both ontologies, resulting in a single thesaurus. This means that when new synonym terms are derived, they can be based on synonym terms inferred from the same ontology, or from both ontologies. We name these two options as intra- and inter-ontology synonym derivation, respectively. This approach is described in Algorithm 1, where creating a thesaurus T is achieved by finding the overlapping portion of the names of each class c in an ontology O, and inferring a synonym relation between the non-overlapping portion. Extension of synonyms through derivation is based on the computation of

Algorithm 1. Create thesaurus from name properties

```
input: O
  T ← ∅
  for each c ∈ O do
    names ← c.getNames()
    for each n1 ∈ names do
      for each n2 ∈ names do
        common_term ← n1.overlap(n2)
        n1_synonym_term ← n1.remove(common_term)
        n2_synonym_term ← n2.remove(common_term)
        T.add(n1_synonym_term, n2_synonym_term)
      end for
    end for
  end for
  return T
```

Algorithm 2. Extend synonyms based on derivation

```
input: O, max
  //get all n-grams of all ontology names with sizes [1,max]
  names ← O.getNames()
  ngrams ← names.getNgrams(max)
  for each ngram ∈ ngrams do
    names_n ← names.contain(ngram)
    thes_n ← T.get(ngram) //thesaurus entries that match the n-gram
    for each name ∈ names_n do
      class ← O.getClass(name)
      for each t ∈ thes_n do
        new_name ← name.replace(ngram, t)
        class.addNewName(new_name)
      end for
    end for
  end for
```

all ontology classes' names n-grams, which can then be replaced by appropriate thesaurus entries. This approach is described in Algorithm 2.

We also propose another approach to create new synonyms that is based on removing common words (i.e., words that convey little information) from the beginning or the end of names, such as 'structure' in 'spinal nerve structure'. To identify common words, we compute the evidence content for each word present in ontology names, according to the inverse logarithm of its frequency [16], then select those below a given evidence content threshold. Then for each name, we create a new synonym where leading and trailing common words are removed. We have called this approach Common Word Removal Synonym Extension (CWSynExt), and describe it in Algorithm 3.

Coupled with this strategy, we have implemented a weighting method, where the weight of the newly created synonym is equal to the weight of the original name multiplied by a confidence factor, which is given by the total evidence content of the synonym divided by the total evidence content of the original name. Thus, the lower the total evidence content of the removed words is, the closer the synonym captures the information conveyed by the original name and the higher will be its confidence factor.

Algorithm 3. Extend synonyms based on common word removal

```
input: O
  for each name ∈ O.getNames() do
    new_name = name
    //checks leading words
    for each word ∈ new_name do
      while word ∈ common_words do
        new_name ← new_name.remove(word)
      end while
    end for
    //checks trailing words
    for each word ∈ new_name.reversed() do
      while word ∈ common_words do
        new_name ← new_name.remove(word)
      end while
    end for
    if new_name ≠ name then
      class ← O.getClass(name)
      class.addNewName(new_name)
    end if
  end for
```

3.3 Ontology Lexicon Extension Using External Ontologies

Given the abundance of biomedical ontologies with overlapping domains, it makes sense to capitalize on correspondences to a mediating ontology to help derive the final correspondences between the ontologies to align [17]. A mediating ontology can be particularly helpful if it contains a large number of synonyms. This approach of matching a mediating ontology to each ontology and then use these results to arrive at the final alignment has been successfully used by several ontology matching systems in the biomedical domain [9, 8].

However, many biomedical ontologies encode cross-references to external ontologies, which represent relationships between classes belonging to distinct ontologies. To the best of our knowledge these have never been explicitly explored by ontology matching systems. These cross-references can be used to extend the lexicon of the ontologies being matched, by adding the name properties of the cross-referenced class to the class of the ontology being matched. For instance, the UBERON ontology encodes cross-references to the Mouse Anatomy ontology, which means that all names and synonyms of an UBERON class that references a Mouse Anatomy class can be added to its synset. This strategy bypasses the need to rely on a lexical matching between the ontologies, since the transference of the names is based on the ontology defined properties.

4 Evaluation

To evaluate our approaches that use synonyms in biomedical ontology matching we use the AgreementMakerLight system [18], a lightweight framework based on the AgreementMaker system [19], which has been optimized to handle the matching of larger ontologies. AgreementMakerLight supports a wide variety of matching methods, called *matchers*, which can be used in series or parallel such that the results from several matching algorithms can be combined into a single

final result, and where correspondences are filtered by a similarity threshold. It is based on the same approaches of AgreementMaker, which have achieved top results in OAEI tracks in several years [20–22].

To remain focused on lexical approaches to ontology matching, we restrict our evaluation to two matchers that are based on the pairwise comparison of ontology classes: a name-based matcher and a word-based matcher. These matchers correspond to commonly used techniques, which are used across several other ontology matching systems (e.g., GOMMA, LogMap and YAM++ [23]). The name-based matcher (NM) consists of a straightforward comparison of the full labels or synonyms of ontology classes. The word-based matcher (WM) relies on the comparison of the words belonging to the labels or synonyms of classes through a weighted Jaccard similarity based on the evidence content of words within ontologies [18]. Although we implement our approaches as extensions to the AML framework, they are independent from it and can be used with any ontology matching system that uses lexical-based matching. To maintain further the independence of our approaches from any specific configurations of AML, we choose to combine the results of matchers through a simple join, and select them based on an empirically chosen threshold of 0.6.

We test our approaches on four matching tasks proposed by OAEI: (1) Mouse Anatomy (MA) - NCI Human Anatomy (NCI_Human), (2) FMA - NCI; (3) FMA-SNOMED; (4) NCI-SNOMED. The first task corresponds to the anatomy track, and the remaining three belong to the *large biomed* track. In the *large biomed* tasks we are only aligning small overlapping fragments, which is one of the tasks supported by OAEI. This means that the portion of FMA being aligned in task 2 is not the same one that is being aligned in task 3. The same applies to NCI and SNOMED. The reference alignment used in the anatomy track was manually created and has been extensively tested. For the *large biomed* track the existing reference alignment is a silver standard based on mappings encoded in UMLS, a biomedical terminology resource [24].

5 Results and Discussion

We first evaluate an approach that uses a ranking and weighting strategy for name properties. Table 2 shows the impact on F-measure when using our proposed default ranking and weighting strategy with an interval of 0.05, in combination with two matching approaches: NM by itself, or combined with WM. Weighting of name properties has a very noticeable impact on the alignment of the mouse and human anatomies, however that impact is much reduced in the other three matching tasks. Based on these results, and since the computational cost for this strategy is quite low, we incorporate the ranking and weighting approach into our other approaches as well.

Our second approach extends the number of synonyms in ontologies either through a synonym derivation technique based on internal ontology synonyms or on the removal of common words (described in Algorithms 1, 2, and 3). These we consider to be internal synonym extension strategies, since they only use

Table 2. Ranking and weighting synonym properties

Matchers	MA-NCI_Human	FMA-NCI	FMA-SNOMED	SNOMED-NCI
Standard AML				
NM	0.819	0.826	0.411	0.689
NM-WM	0.829	0.838	0.586	0.732
Ranking & Weighting				
NM	0.825	0.826	0.412	0.689
NM-WM	0.862	0.840	0.586	0.732

NM: name-based matcher; WM: word-based matcher

Comparison of the F-measure obtained in all four tasks when using the ranking and weighting strategy with two different matching approaches, one based on matching the full name and the other also considering word matches.

information contained in the ontologies that are being aligned. However, external ontologies can also be used to increase the number of synonyms through the transference of names from cross-referenced classes. As a source for cross-references we use the UBERON ontology, which encodes direct cross-references to the mouse and human anatomies, as well as NCI. Figure 1 shows the increase in number of name properties in each ontology after synonym extension. The number of new name properties created by intra- and inter-ontology synonym derivation is closely tied to the original number of synonyms (see Table 1), therefore for SNOMED the use of intra-ontology synonym extension does not lead to a noticeable increase in number of name properties, since there are very few synonyms to leverage on to create the internal thesaurus. However, when ontologies have very frequent words in their terminology, the number of synonyms created by the common word removal approach increases. This is clearly exemplified by SNOMED, where the existence of many names with common words such as 'structure' (e.g., 'structure of hair of trunk', 'portal vein structure' and 'spinal nerve structure') results in the creation of many more synonyms.

To test the impact of synonym extension we couple it with the NM matcher. Both intra- and inter-ontology synonym derivation can lead to a high number of erroneous names, however when used with the NM matcher these issues are circumvented since a single match between two names is enough to map two classes and the presence of erroneous words in the names has no impact. Given the low impact intra- and inter-ontology synonym derivation has on SNOMED's terminology, we would expect a reduced impact of these strategies on the matching performance of SNOMED alignments, particularly when using the inter-ontology approach. Indeed, in the last two tasks (see Table 3), FMA-SNOMED and SNOMED-NCI have an equivalent or reduced performance when using this approach. In particular, extending SNOMED with inter-ontology synonyms leads to a marked drop in precision. On the other hand, for the alignment of the mouse and human anatomies, synonym derivation improves performance through an increase in recall, particularly for the intra-ontology approach where recall increases by 7.5%. In FMA-NCI, there is also an improvement, though not

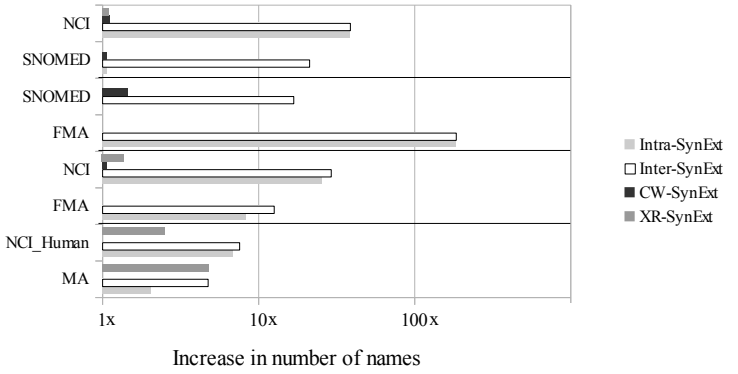


Fig. 1. Increase in number of names after synonym extension approaches for each ontology in each task

(Intra-: intra-ontology; Inter-: inter-ontology; CW-: common word removal; XR-: cross-references to UBERON; SynExt: synonym extension)

as marked, with recall increasing by 1.7%. The common word removal synonym extension approach has little to no impact on the MA-NCI_Human and FMA-NCI alignments, but has a considerable impact on FMA-SNOMED, where it increases recall by more than 40%, increasing F-measure from 41.2% to 74.5%. This is due to the fact that removing the common words in SNOMED names results in direct matches to several FMA classes. This effect is less noticeable in SNOMED-NCI, but it still increases recall by nearly 5%.

Our third approach is based on exploring external ontologies that contain cross-references to the ontologies that are to be matched, or whose domains are closely related. In this evaluation we use three ontologies as external resources: UBERON, FMA and NCI. FMA and NCI versions correspond to the full ontologies (obtained from OBO, not from OAEI). Table 4 presents the results of several distinct matching strategies that use these external ontologies. Using a combination of NM and a mediating matcher (MM) based on NM to FMA (NM-MM), results in a better performance in the mouse and human anatomies as well as in SNOMED-NCI. The same strategy using NCI only impacts SNOMED-NCI results. However, when UBERON is used, there is a marked improvement in both MA-NCI_Human and FMA-NCI, which is due to MA, NCI_Human, FMA and UBERON sharing the same domain (anatomy).

UBERON encodes cross-references to MA, NCI, SNOMED and FMA. However, the cross-references are established using alphanumeric identifiers, which are unavailable in the OAEI versions of FMA and SNOMED. Consequently, we have only explored the cross-references to MA and NCI. For the MA-NCI_Human, given that UBERON encodes cross-references to both ontologies it is possible to create an alignment based solely on them (XRM). This has an F-measure of 91.7%, which is higher than any of the other approaches tested so far. A combination with NM further increases F-measure up to 92.6%. However, the

Table 3. Impact of internal synonym extension approaches on matching performance

	No SynExt	Inter-SynExt	Intra-SynExt	CW-SynExt
	MA-NCI-human			
Precision	0.985	0.983	0.966	0.985
Recall	0.691	0.709	0.766	0.691
F-measure	0.825	0.835	0.860	0.825
	FMA-NCI			
Precision	0.945	0.936	0.939	0.944
Recall	0.723	0.736	0.74	0.723
F-measure	0.826	0.83	0.834	0.827
	FMA-SNOMED			
Precision	0.953	0.926	0.945	0.897
Recall	0.178	0.182	0.180	0.618
F-measure	0.412	0.411	0.413	0.745
	NCI-SNOMED			
Precision	0.97	0.888	0.965	0.967
Recall	0.489	0.477	0.497	0.537
F-measure	0.689	0.651	0.693	0.721

(Intra-: intra-ontology; Inter-: inter-ontology; CW-: common word removal; SynExt: synonym extension)

cross-references can also be explored to extend the name properties of classes and then be used on an NM matching approach (NM-XR-SynExt), pushing F-measure up by another 0.9%. Combining this approach with the more complex WM results in an F-measure of 93.7% (NM-XR-SynExt-WM). The synonym extension that is based on cross-references can also be used in the NCI matching tasks, which yields the best performance we obtained for FMA-NCI, 86.4%, but has no impact on SNOMED-NCI. This is likely due to the fact that the NCI fragment in FMA-NCI belongs to the anatomy domain (the same as UBERON), whereas the SNOMED and NCI fragments of NCI-SNOMED do not.

The overall very positive success of exploring cross-references, both for direct matching and for synonym extension, clearly demonstrates the untapped potential of these ontology properties.

To complete our evaluation we present a table with the comparison of our best results with the best results obtained by OAEI 2012 competitors in each task (see Table 5). For simplicity we name the integration of our approaches into AML as AMLnym. Our best results are obtained using two distinct strategies: for the MA-NCI_Human and FMA-NCI tasks the two lexical matchers (name-based and word-based) are coupled with the synonym extension derived from UBERON cross-references (NM-WM-XR-SynExt), whereas for the FMA-SNOMED and SNOMED-NCI they are coupled with the common word removal synonym extension (NM-WM-CW-SynExt). The only task where we surpass the best OAEI competitor is the MA-NCI_Human, where the use of cross-references to extend the name properties has a positive impact on performance, with 93.7% in F-measure, which is 1.4% higher than the top ranked system GOMMA-bk.

Table 4. Using External Ontologies through cross-references and matching

Matchers	MA-NCI_H.	FMA-NCI	FMA-SNM	SNM-NCI	Ext. Ont.
NM-MM	0.837	0.826	0.412	0.691	FMA
	0.826	0.827	0.412	0.691	NCI
	0.910	0.849	0.412	0.690	UBERON
XRM	0.917	N.A.	N.A.	N.A.	
+NM	0.926	N.A.	N.A.	N.A.	
NM-XR-SynExt	0.935	0.864	N.A.	0.690	
+MM	0.936	N.A.	N.A.	N.A.	
+WM	0.937	N.A.	N.A.	N.A.	

Comparison of the F-measure obtained when using different matching techniques and external ontologies to support matching. (XRM: cross-references matcher; MM: mediating matcher; WM: word-based matcher; XR-SynExt: cross-references based synonym extension)

Table 5. Comparison of our approaches with the best OAEI 2012 competitors in each task.

			MA-NCI_Human	FMA-NCI	FMA-SNOMED	NCI-SNOMED
OAEI 2012	AML+Nym		NM-WM-XR-SynExt		NM-WM-CW-SynExt	
		P	0.957	0.940	0.870	0.925
		R	0.917	0.802	0.670	0.589
		F	0.937	0.869	0.763	0.738
	UMLS		GOMMA-bk	GOMMA-bk	GOMMA-bk	LogMapnoe
		P	0.917	0.914	0.826	0.893
		R	0.928	0.922	0.912	0.659
		F	0.923	0.918	0.886	0.758
	no UMLS			GOMMA	GOMMA	LogMapLt
		P		0.945	0.834	0.938
		R		0.856	0.377	0.560
		F		0.898	0.520	0.701

Comparison of the performance obtained by our approaches (AML+Nym) with the best competitors in OAEI 2012 (GOMMA and LogMap) with and without the use of UMLS as an external resource (P:Precision; R:Recall; F:F-measure; NM: name-based matcher; NM: name-based matcher; WM: word-based matcher; XR-SynExt: cross-references based synonym extension; CW-SynExt: common word removal synonym extension)

For the other three tasks our results are below those obtained by the leading systems. However, both GOMMA-bk and LogMapnoe use UMLS as an external resource. Since the reference alignment is a silver standard based on UMLS, using the same resource is a biased approach that clearly results in improved performance. Considering this, we also include in the table the results obtained

by those systems when using their less elaborate variants, which do not use UMLS. In FMA-NCI we still remain below GOMMA's results by 2.9%, but in the remaining tasks our approaches have a better performance, with an advantage of 24.3% in FMA-SNOMED over GOMMA and 3.7% in SNOMED-NCI over LogMapLt. However, it is important to note that GOMMA and LogMapLt differ from their more complete variants in more than just the use of UMLS, which can also explain part of the drop in performance.

6 Conclusions

We have presented three novel approaches for a better use of an ontology's terminological properties within ontology matching tasks. These approaches capitalize on biomedical ontology properties such as a rich terminology, with several synonyms of different kinds being encoded, as well as the existence of related ontologies with overlapping domains.

Our first approach distinguished between different name properties by assigning to them weights that reflect their closeness in meaning to the main name. Our results demonstrate the success of this strategy, which resulted in an increase in performance for several terminological-based matchers. Furthermore, we have shown that it is possible to extend the number of name properties of an ontology through two synonym derivation techniques, one which explores the reflexive property of synonyms to infer synonymy between words or multi-word terms that belong to synonym labels, and used these terms to compose new synonym labels, and another based on common word removal. In many cases these approaches increase the performance of name and word-based matchers up to competitive levels with more complex strategies based on external resources and structural approaches. However, the success of the synonym derivation technique based on synonym terms depends on the existence of synonyms encoded by the ontologies, which is why it is less suited for ontologies with few synonyms such as SNOMED. The synonym derivations techniques can be also be used for ontology extension, since they are able to add novel synonyms to an ontology. Ontology extension in the biomedical domain is a budding field [25, 26], for which ontology matching has been identified as a crucial technique [27–29]. Finally, our third approach consisted in using ontologies with cross-references to the ontologies being aligned. This was shown to have a high impact on matching performance, both when the strategy was used to directly produce matches, and when it was used to extend the number of synonyms within ontologies.

The application of these approaches to OAEI tasks demonstrated the impact they can have on ontology matching performance. In the anatomy track, our results were better than those obtained by the best OAEI 2012 participant. In the three tasks of the *large biomed* track, our strategies proved insufficient to place above the leading systems. However, these systems benefit strongly from using UMLS as an external resource, and also from structural and logic-based strategies. When we compare our results with simpler versions of the leading systems that do not use these additional strategies, our approaches produce the

best results in two out of three tasks. These results lead us to believe that the integration of our approaches in more complex matching strategies, using both structural and logic-based matchers will lead to an improvement of the current state of the art in biomedical ontology matching.

Furthermore, our results demonstrate that when there is an adequate external resource that links both ontologies, using it as a source for synonym extension can strongly improve matching performance. Ascertaining if an external resource is relevant for a matching task is then a relevant question, which we will address in future work. We also hope to address the extension of our synonym derivation technique to other kinds of relations such as hypernymy and holonymy.

We have demonstrated the importance of an adequate consideration of terminological properties in ontology matching, specifically of distinguishing between different synonym properties and of extending synonyms based both on ontology internal knowledge and on references to external resources. Our novel approaches will become increasingly relevant as ontologies grow and become more refined, defining more synonyms through distinct properties. We envision that the next step in exploring synonyms in biomedical ontology matching will include finding other kinds of relations, not just equivalence, so as to enable linking different entities such as diseases, symptoms, genes, anatomical structures, phenotypes and organisms, in a true biomedical Semantic Web.

Acknowledgments. DF, CP, ES and FMC were funded by the Portuguese FCT through the SOMER project (PTDC/EIA-EIA/119119/2010) and the multi-annual funding program to LASIGE. IFC and CS were partially supported by NSF Awards IIS-0812258, IIS-1143926, and IIS-1213013. IFC was also supported by a University of Illinois Scholar Award, by a UIC Area of Excellence Award, and by a UIC-IPCE Civic Engagement Research Fund Award.

References

1. Rubin, D.L., Shah, N.H., Noy, N.F.: Biomedical ontologies: a functional perspective. Briefings in Bioinformatics 9(1), 75–90 (2008)
2. Euzenat, J., Shvaiko, P.: Ontology matching, vol. 18. Springer, Berlin (2007)
3. Smith, B., Ashburner, M., Rosse, C., Bard, J., Bug, W., Ceusters, W., Goldberg, L., Eilbeck, K., Ireland, A., Mungall, C., et al.: The OBO Foundry: coordinated evolution of ontologies to support biomedical data integration. Nature Biotechnology 25(11), 1251–1255 (2007)
4. Noy, N., Shah, N., Whetzel, P., Dai, B., Dorf, M., Griffith, N., Jonquet, C., Rubin, D., Storey, M., Chute, C., et al.: Bioportal: ontologies and integrated data resources at the click of a mouse. Nucleic Acids Research 37(suppl. 2), 170–173 (2009)
5. Euzenat, J., Meilicke, C., Stuckenschmidt, H., Shvaiko, P., Trojahn, C.: Ontology alignment evaluation initiative: six years of experience. Journal on Data Semantics XV, 158–192 (2011)
6. Ghazvinian, A., Noy, N., Musen, M.: Creating mappings for ontologies in biomedicine: Simple methods work. In: AMIA Annual Symposium (AMIA 2009), vol. (2) (2009)

7. Lambrix, P., Tan, H.: SAMBO - system for aligning and merging biomedical ontologies. Web Semantics: Science, Services and Agents on the World Wide Web 4(3), 196–206 (2006)
8. Groß, A., Hartung, M., Kirsten, T., Rahm, E.: GOMMA results for OAEI 2012. In: Ontology Matching Workshop. International Semantic Web Conference 2012 (2012)
9. Cruz, I.F., Stroe, C., Caimi, F., Fabiani, A., Pesquita, C., Couto, F.M., Palmonari, M.: Using AgreementMaker to Align Ontologies for OAEI 2011. In: ISWC International Workshop on Ontology Matching (OM). CEUR Workshop Proceedings, vol. 814, pp. 114–121 (2011)
10. Cruz, I.F., Palandri Antonelli, F., Stroe, C.: AgreementMaker: Efficient Matching for Large Real-World Schemas and Ontologies. PVLDB 2(2), 1586–1589 (2009)
11. Xu, P., Tao, H., Zang, T.: Alignment Results of SOBOM for OAEI 2009. In: The 4th International Workshop on Ontology Matching at ISWC 2009 (2009)
12. Jiménez-Ruiz, E., Grau, B.C., Zhou, Y.: LogMap 2.0: towards logic-based, scalable and interactive ontology matching. In: Proc. of the 4th International Workshop on Semantic Web Applications and Tools for the Life Sciences, pp. 45–46 (2011)
13. Aguirre, J.L., Eckert, K., Euzenat, J., Ferrara, A., van Hage, W.R., Hollink, L., Meilicke, C., Nikolov, A., Ritze, D., Scharffe, F., Shvaiko, P., Sváb-Zamazal, O., dos Santos, C.T., Jiménez-Ruiz, E., Grau, B.C., Zapilko, B.: Results of the Ontology Alignment Evaluation Initiative 2012. In: ISWC International Workshop on Ontology Matching (OM). CEUR Workshop Proceedings, vol. 946, pp. 73–115 (2012)
14. Cruz, I.F., Sunna, W., Makar, N., Bathala, S.: A Visual Tool for Ontology Alignment to Enable Geospatial Interoperability. Journal of Visual Languages and Computing 18(3), 230–254 (2007)
15. Morant, A.: Extending and optimizing an ontology matching system (2011)
16. Couto, F., Silva, M., Coutinho, P.: Finding genomic ontology terms in text using evidence content. BMC Bioinformatics 6(suppl. 1), S21 (2005)
17. Gross, A., Hartung, M., Kirsten, T., Rahm, E.: Mapping composition for matching large life science ontologies. In: ICBO (2011)
18. Faria, D., Pesquita, C., Santos, E., Palmonari, M., Cruz, I.F., Couto, F.M.: The AgreementMakerLight Ontology Matching System. In: ODBASE (2013)
19. Cruz, I.F., Sunna, W.: Structural Alignment Methods with Applications to Geospatial Ontologies. Transactions in GIS, Special Issue on Semantic Similarity Measurement and Geospatial Applications 12(6), 683–711 (2008)
20. Euzenat, J., Ferrara, A., Hollink, L., Isaac, A., Joslyn, C., Malaisé, V., Meilicke, C., Nikolov, A., Pane, J., Sabou, M., Scharffe, F., Shvaiko, P., Spiliopoulos, V., Stuckenschmidt, H., Sváb-Zamazal, O., Svátek, V., dos Santos, C.T., Vouros, G.A., Wang, S.: Results of the Ontology Alignment Evaluation Initiative 2009. In: ISWC International Workshop on Ontology Matching (OM). CEUR Workshop Proceedings, vol. 551, pp. 73–126 (2009)
21. Euzenat, J., Ferrara, A., Meilicke, C., Pane, J., Scharffe, F., Shvaiko, P., Stuckenschmidt, H., Sváb-Zamazal, O., Svátek, V., dos Santos, C.T.: Results of the Ontology Alignment Evaluation Initiative 2010. In: ISWC International Workshop on Ontology Matching (OM). CEUR Workshop Proceedings, vol. 689, pp. 85–117 (2010)
22. Euzenat, J., Ferrara, A., van Hage, W.R., Hollink, L., Meilicke, C., Nikolov, A., Ritze, D., Scharffe, F., Shvaiko, P., Stuckenschmidt, H., Sváb-Zamazal, O., dos Santos, C.T.: Results of the Ontology Alignment Evaluation Initiative 2011. In: ISWC International Workshop on Ontology Matching (OM). CEUR Workshop Proceedings, vol. 814, pp. 85–113 (2011)

23. Ngo, D., Bellahsene, Z.: Yam++: A multi-strategy based approach for ontology matching task. In: ten Teije, A., Völker, J., Handschuh, S., Stuckenschmidt, H., d'Acquin, M., Nikolov, A., Aussenac-Gilles, N., Hernandez, N. (eds.) EKAW 2012. LNCS, vol. 7603, pp. 421–425. Springer, Heidelberg (2012)
24. Bodenreider, O.: The Unified Medical Language System (UMLS): integrating biomedical terminology. Nucleic Acids Res. 32 (Database issue) (January 2004)
25. Pesquita, C., Couto, F.: Predicting the extension of biomedical ontologies. PLOS Computational Biology 8(9), e1002630 (2012)
26. Wächter, T., Schroeder, M.: Semi-automated ontology generation within OBO-Edit. Bioinformatics 26, 88–96 (2010)
27. Novácek, V., Laera, L., Handschuh, S., Davis, B.: Infrastructure for dynamic knowledge integration–automated biomedical ontology extension using textual resources. Journal of Biomedical Informatics 41(5), 816–828 (2008)
28. Pesquita, C., Stroe, C., Cruz, I.F., Couto, F.: BLOOMS on AgreementMaker: Results for OAEI 2010. In: ISWC International Workshop on Ontology Matching (OM). CEUR Workshop Proceedings, vol. 689, pp. 135–141 (2010)
29. Pesquita, C.: Automated extension of biomedical ontologies. PhD thesis, University of Lisbon (2012)

Knowledge Graph Identification

Jay Pujara[1], Hui Miao[1], Lise Getoor[1], and William Cohen[2]

[1] Dept of Computer Science, University of Maryland, College Park, MD 20742
{jay,hui,getoor}@cs.umd.edu
[2] Machine Learning Dept, Carnegie Mellon University, Pittsburgh, PA 15213
wcohen@cs.cmu.edu

Abstract. Large-scale information processing systems are able to extract massive collections of interrelated facts, but unfortunately transforming these candidate facts into useful knowledge is a formidable challenge. In this paper, we show how uncertain extractions about entities and their relations can be transformed into a *knowledge graph*. The extractions form an *extraction graph* and we refer to the task of removing noise, inferring missing information, and determining which candidate facts should be included into a knowledge graph as *knowledge graph identification*. In order to perform this task, we must reason jointly about candidate facts and their associated extraction confidences, identify coreferent entities, and incorporate ontological constraints. Our proposed approach uses probabilistic soft logic (PSL), a recently introduced probabilistic modeling framework which easily scales to millions of facts. We demonstrate the power of our method on a synthetic Linked Data corpus derived from the MusicBrainz music community and a real-world set of extractions from the NELL project containing over 1M extractions and 70K ontological relations. We show that compared to existing methods, our approach is able to achieve improved AUC and F1 with significantly lower running time.

1 Introduction

The web is a vast repository of knowledge, but automatically extracting that knowledge at scale has proven to be a formidable challenge. Recent evaluation efforts have focused on automatic knowledge base population [1,2], and many well-known broad domain and open information extraction systems exist, including the Never-Ending Language Learning (NELL) project [3], OpenIE [4], and efforts at Google [5], which use a variety of techniques to extract new knowledge, in the form of facts, from the web. These facts are interrelated, and hence, recently this extracted knowledge has been referred to as a knowledge graph [6].

A key challenge in producing the knowledge graph is incorporating noisy information from different sources in a consistent manner. Information extraction systems operate over many source documents, such as web pages, and use a collection of strategies to generate candidate facts from the documents, spanning syntactic, lexical and structural features of text. Ultimately, these extraction systems produce candidate facts that include a set of entities, attributes of these entities, and the relations between these entities which we refer to as the extraction graph. However errors in the extraction process introduce inconsistencies in

H. Alani et al. (Eds.): ISWC 2013, Part I, LNCS 8218, pp. 542–557, 2013.

the extraction graph, which may contain duplicate entities and violate key ontological constraints such as subsumption, mutual exclusion, inverse, domain and range constraints. Such noise obscures the true knowledge graph, which captures a consistent set of entities, attributes and relations.

Our work infers the knowledge graph from the extraction graph generated by an information extraction system. We demonstrate that the errors encountered by information extraction systems require jointly reasoning over candidate facts to construct a consistent knowledge graph. Our approach performs entity resolution, collective classification and link prediction while also enforcing global constraints on the knowledge graph, a process which we refer to as *knowledge graph identification*.

In order to implement knowledge graph identification, we use probabilistic soft logic (PSL) [7], a recently introduced framework for reasoning probabilistically over continuously-valued random variables. PSL provides many advantages: models are easily defined using declarative rules with first-order logic syntax, continuously-valued variables provide a convenient representation of uncertainty, weighted rules and weight learning capture the importance of model rules, and advanced features such as set-based aggregates and hard constraints are supported. In addition, inference in PSL is a convex optimization that is highly scalable allowing us to handle millions of facts in minutes.

We develop a PSL model for knowledge graph identification that both captures probabilistic dependencies between facts and enforces global constraints between entities and relations. Through this model, we define a probability distribution over interpretations - or truth value assignments to facts - each of which corresponds to a possible knowledge graph. By performing inference using the extraction graph and an ontology, we are able to find the most probable knowledge graph. We establish the benefits of our approach on two large datasets: a synthetic dataset derived from the MusicBrainz community and ontological relationships defined in the Music Ontology as well as noisy extractions from NELL, a large-scale operational knowledge extraction system.

Our contributions in this work are 1) formulating the knowledge graph identification problem that supports reasoning about multiple, uncertain extractor sources in the presence of ontological constraints; 2) solving knowledge graph identification efficiently with convex optimization using PSL; and 3) demonstrating the power of knowledge graph identification by presenting results on benchmark datasets that are superior to state-of-the-art methods and generating massive knowledge graphs on the scale of minutes that are infeasible to compute in competing systems.

2 Related Work

Early work on the problem of jointly identifying a best latent KB from a collection of noisy facts was considered by Cohen et al. [8], however they considered only a small subset of KB errors. More recently, Jiang et al. [9] perform knowledge base refinement at a broader scope by using an ontology to relate candidate extractions and exploring many different modeling choices with Markov Logic

Networks (MLNs) [10]. Jiang et al. provide a crisp codification of ontological constraints and candidate facts found in a knowledge base as rules in first-order logic, contributing an attractive abstraction for knowledge bases which we adopt in our modeling. However, the choice of MLNs as a modeling framework comes with certain limitations. In MLNs, all logical predicates must take Boolean truth values, making it difficult to incorporate the confidence values. Moreover, the combinatorial explosion of Boolean assignments to random variables makes inference and learning in MLNs intractable optimization problems. Jiang et al. surmount these obstacles with a number of approximations and demonstrate the utility of joint reasoning in comparison to a baseline that considers each fact independently. By using PSL we can avoid these representational and scalability limitations, and we build on and improve the model of Jiang et al. by including multiple extractors in our model and reasoning about co-referent entities.

Other research has used relevant techniques for problems related to knowledge graph identification. Namata et al. [11] introduced the problem of graph identification to uncover the true graph from noisy observations through entity resolution, collective classification, and link prediction. However, Namata's approach considered these tasks iteratively and could not easily support logical constraints such as those found in an ontology. Memory et al. [12] also use PSL to resolve confounding evidence. Their model performs graph summarization across multiple ontologies and uses inference only for inferring missing links. Work by Yao et al. [13] employs joint reasoning at the extractor level by using conditional random fields to learn selectional preferences for relations.

3 Motivation: Knowledge Graph Identification

In this work, we represent the candidate facts from an information extraction system as a knowledge graph where entities are nodes, categories are labels associated with each node, and relations are directed edges between the nodes. Information extraction systems can extract such candidate facts, and these extractions can be used to construct an extraction graph. Unfortunately, the extraction graph is often incorrect, with errors such as spurious and missing nodes and edges, and missing or inaccurate node labels. Our approach, knowledge graph identification (KGI) combines the tasks of entity resolution, collective classification and link prediction mediated by rules based on ontological information. We motivate the necessity of our approach with examples of challenges taken from a real-world information extraction system, the Never-Ending Language Learner (NELL) [3].

Entity extraction is a common problem: many textual references that initially look different may refer to the same real-world entity. For example, NELL's knowledge base contains candidate facts involving the entities "kyrghyzstan", "kyrgzstan", "kyrgystan", "kyrgyz republic", "kyrgyzstan", and "kyrgistan" which are all variants or misspellings of the country Kyrgyzstan. In the extracted knowledge graph, these incorrectly correspond to different nodes. Our approach uses *entity resolution* to determine co-referent entities in the knowledge graph, producing a consistent set of labels and relations for each resolved node.

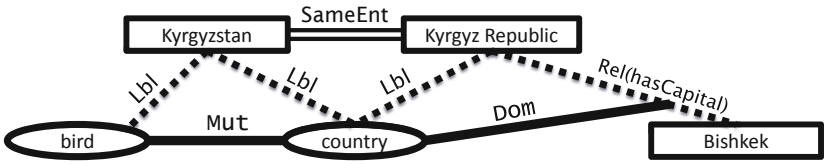


Fig. 1. An illustration of the example showing how knowledge graph identification can resolve conflicting information in an extraction graph. Entities are shown in rectangles, dotted lines represent uncertain information, solid lines show ontological constraints and double lines represent co-referent entities found with entity resolution.

Another challenge in knowledge graph construction is inferring labels consistently. For example, NELL's extractions assign Kyrgyzstan the labels "country" as well as "bird." Ontological information suggests that an entity is very unlikely to be both a country and a bird at the same time. Using the labels of related entities in the knowledge graph can allow us to determine the correct label of an entity. Our approach uses *collective classification* to label nodes in manner which takes into account ontological information and neighboring labels.

A third problem commonly encountered in knowledge graphs is determining the relationships between entities. NELL also has many facts relating the location of Kyrgyzstan to other entities. These candidate relations include statements that Kyrgyzstan is located in Kazakhstan, Kyrgyzstan is located in Russia, Kyrgyzstan is located in the former Soviet Union, Kyrgyzstan is located in Asia, and that Kyrgyzstan is located in the US. Some of these possible relations are true, while others are clearly false and contradictory. Our approach uses *link prediction* to predict edges in a manner which takes into account ontological information and the rest of the inferred structure.

Refining an extraction graph becomes even more challenging as we consider the interaction between the predictions and take into account the confidences we have in the extractions. Figure 1 illustrates such a complex example. As mentioned earlier, NELL's ontology includes the constraint that the labels "bird" and "country" are mutually exclusive. Reasoning collectively allows us to resolve which of these two labels is more likely to apply to Krygyzstan. For example, NELL is highly confident that the Kyrgyz Republic has a capital city, Bishkek. The NELL ontology specifies that the domain of the relation "hasCapital" has label "country." Entity resolution allows us to infer that "Kyrgyz Republic" refers to the same entity as "Kyrgyzstan." Deciding whether Kyrgyzstan is a bird or a country now involves a prediction where we include the confidence values of the corresponding "bird" and "country" facts from co-referent entities, as well as collective features from ontological relationships of these co-referent entities, such as the confidence values of the "hasCapital" relations. We refer to this process of inferring a knowledge graph from a noisy extraction graph as knowledge graph identification. Unlike earlier work on graph identification and knowledge base refinement, we use a very different probabilistic framework, PSL, allowing us to jointly infer a knowledge graph while incorporating extractor confidence values and supporting a rich collection of ontological constraints.

4 Background: Probabilistic Soft Logic

Probabilistic soft logic (PSL) [7,14] is a recently-introduced framework which allows users to specify rich probabilistic models over continuous-valued random variables. Like other statistical relational learning languages such as Markov Logic Networks (MLNs), it uses first-order logic to describe features that define a Markov network. In contrast to other approaches, PSL employs continuous-valued random variables rather than binary variables and casts most probable explanation (MPE) inference as a convex optimization problem that is significantly more efficient to solve than its combinatorial counterpoint (polynomial vs. exponential).

A PSL model is composed of a set of weighted, first-order logic rules, where each rule defines a set of features of a Markov network sharing the same weight. Consider the formula

$$\mathrm{P}(A,B)\tilde{\wedge}\mathrm{Q}(B,C) \overset{w}{\Rightarrow} \mathrm{R}(A,B,C)$$

which is an example of a PSL rule. Here w is the weight of the rule, A, B, and C are universally-quantified variables, and P, Q and R are predicates. A *grounding* of a rule comes from substituting constants for universally-quantified variables in the rule's atoms. In this example, assigning constant values a, b, and c to the respective variables in the rule above would produce the ground atoms P(a,b), Q(b,c), R(a,b,c). Each ground atom takes a soft-truth value in the range $[0, 1]$.

PSL associates a numeric *distance to satisfaction* with each ground rule that determines the value of the corresponding feature in the Markov network. The distance to satisfaction is defined by treating the ground rule as a formula over the ground atoms in the rule. In particular, PSL uses the *Lukasiewicz t-norm* and *co-norm* to provide a relaxation of the logical connectives, AND ($\wedge$), OR($\vee$), and NOT($\neg$), as follows (where relaxations are denoted using the $\sim$ symbol over the connective):

$$p\tilde{\wedge}q = \max(0, p+q-1)$$
$$p\tilde{\vee}q = \min(1, p+q)$$
$$\tilde{\neg}p = 1-p$$

This relaxation coincides with Boolean logic when p and q are in $\{0, 1\}$, and provides a consistent interpretation of soft-truth values when p and q are in the numeric range $[0, 1]$.

A PSL program, $\mathbf{\Pi}$, consisting of a model as defined above, along with a set of facts, F, produces a set of ground rules, R. If I is an interpretation (an assignment of soft-truth values to ground atoms) and r is a ground instance of a rule, then the distance to satisfaction $\phi_r(I)$ of r is $1 - T_r(I)$, where $T_r(I)$ is the soft-truth value from the Lukasiewicz t-norm. We can define a probability distribution over interpretations by combining the weighted degree of satisfaction over all ground rules, R, and normalizing, as follows:

$$f(I) = \frac{1}{Z}\exp\left[-\sum_{r\in R} w_r \phi_r(I)^p\right]$$

Here Z is a normalization constant, w_r is the weight of rule r, and p in $\{1, 2\}$ allows a linear or quadratic combination of rules. Thus, a PSL program (set

of weighted rules and facts) defines a probability distribution from a logical formulation that expresses the relationships between random variables.

MPE inference in PSL determines the most likely soft-truth values of unknown ground atoms using the values of known ground atoms and the dependencies between atoms encoded by the rules, corresponding to inference of random variables in the underlying Markov network. PSL atoms take soft-truth values in the interval $[0, 1]$, in contrast to MLNs, where atoms take Boolean values. MPE inference in MLNs requires optimizing over combinatorial assignments of Boolean truth values. In contrast, the relaxation to the continuous domain greatly changes the tractability of computations in PSL: finding the most probable interpretation given a set of weighted rules is equivalent to solving a convex optimization problem. Recent work from [15] introduces a consensus optimization method applicable to PSL models; their results suggest consensus optimization scales linearly with the number of ground rules in the model.

5 Knowledge Graph Identification Using PSL

Knowledge graphs contain three types of facts: facts about entities, facts about entity labels and facts about relations. We represent entities with the logical predicate Ent(E) and labels with the logical predicate Lbl(E,L) where entity E has label L. Relations are represented with the logical predicate Rel(E_1,E_2,R) where the relation R holds between the entities E_1 and E_2, eg. `R(E1,E2)`.

In knowledge graph identification, our goal is to identify a true set of atoms from a set of noisy extractions. Our method for knowledge graph identification incorporates three components: capturing uncertain extractions, performing entity resolution, and enforcing ontological constraints. We show how we create a PSL program that encompasses these three components, and then relate this PSL program to a distribution over possible knowledge graphs.

5.1 Representing Uncertain Extractions

We relate the noisy extractions from an information extraction system to the above logical predicates by introducing *candidate* predicates, using a formulation similar to [9]. For each candidate entity, we introduce a corresponding predicate, CandEnt(E). Labels or relations generated by the information extraction system correspond to predicates CandLbl(E,L) or CandRel(E_1,E_2,R) in our system. Uncertainty in these extractions is captured by assigning these predicates a soft-truth value equal to the confidence value from the extractor. For example, the extraction system might generate a relation, `hasCapital(kyrgyzstan, Bishkek)` with a confidence of .9, which we would represent as CandRel(`kyrgyzstan`,`Bishkek`, `hasCapital`) and assign it a truth value of .9.

Information extraction systems commonly use many different extraction techniques to generate candidates. For example, NELL produces separate extractions from lexical, structural, and morphological patterns, among others. We represent metadata about the technique used to extract a candidate by using separate predicates for each technique T, of the form CandRel_T and CandLbl_T.

These predicates are related to the true values of attributes and relations we seek to infer using weighted rules.

$$\textsc{CandRel}_T(E_1, E_2, R) \overset{w_{CR-T}}{\Rightarrow} \textsc{Rel}(E_1, E_2, R)$$

$$\textsc{CandLbl}_T(E, L) \overset{w_{CL-T}}{\Rightarrow} \textsc{Lbl}(E, L)$$

Together, we denote the set of candidates, generated from grounding the rules above using the output from the extraction system, as the set $\mathcal{C}$.

5.2 Entity Resolution

While the previous PSL rules provide the building blocks of predicting links and labels using uncertain information, knowledge graph identification employs entity resolution to pool information across co-referent entities. A key component of this process is identifying possibly co-referent entities and determining the similarity of these entities, which we discuss in detail in Section 6. We use the SameEnt predicate to capture the similarity of two entities, for example SameEnt(`kyrgyzstan`, `kyrgz republic`).

To perform entity resolution using the SameEnt predicate we introduce three rules, whose groundings we refer to as $\mathcal{S}$, to our PSL program:

$$\textsc{SameEnt}(E_1, E_2) \tilde{\wedge} \textsc{Lbl}(E_1, L) \overset{w_{EL}}{\Rightarrow} \textsc{Lbl}(E_2, L)$$

$$\textsc{SameEnt}(E_1, E_2) \tilde{\wedge} \textsc{Rel}(E_1, E, R) \overset{w_{ER}}{\Rightarrow} \textsc{Rel}(E_2, E, R)$$

$$\textsc{SameEnt}(E_1, E_2) \tilde{\wedge} \textsc{Rel}(E, E_1, R) \overset{w_{ER}}{\Rightarrow} \textsc{Rel}(E, E_2, R)$$

These rules define an equivalence class of entities, such that all entities related by the SameEnt predicate must have the same labels and relations. The soft-truth value of the SameEnt, derived from our similarity function, mediates the strength of these rules. When two entities are very similar, they will have a high truth value for SameEnt, so any label assigned to the first entity will also be assigned to the second entity. On the other hand, if the similarity score for two entities is low, the truth values of their respective labels and relations will not be strongly constrained. We introduce these rules as weighted rules in the PSL model, where the weights can capture the reliability of the similarity function.

5.3 Enforcing Ontological Constraints

In our PSL program we also leverage rules corresponding to an ontology, the groundings of which are denoted as $\mathcal{O}$. Our ontological rules are based on the logical formulation proposed in [9]. Each type of ontological relation is represented as a predicate, and these predicates represent ontological knowledge of the relationships between labels and relations. For example, the ontological predicates Dom(`hasCapital`, `country`) and Rng(`hasCapital`, `city`) specify that the relation `hasCapital` is a mapping from entities with label `country` to entities with label `city`. The predicate Mut(`country`, `city`) specifies that the labels `country` and `city` are mutually exclusive, so that an entity cannot have both the labels `country` and `city`. We similarly use predicates for subsumption of labels (Sub) and relations(RSub), and inversely-related functions (Inv). To use

this ontological knowledge, we introduce rules relating each ontological predicate to the predicates representing our knowledge graph. We specify seven types of ontological constraints in our experiments using weighted rules:

$$\begin{aligned}
&\textsc{Dom}(R, L) && \tilde{\wedge}\ \textsc{Rel}(E_1, E_2, R) && \overset{w_Q}{\Rightarrow}\ \textsc{Lbl}(E_1, L)\\
&\textsc{Rng}(R, L) && \tilde{\wedge}\ \textsc{Rel}(E_1, E_2, R) && \overset{w_Q}{\Rightarrow}\ \textsc{Lbl}(E_2, L)\\
&\textsc{Inv}(R, S) && \tilde{\wedge}\ \textsc{Rel}(E_1, E_2, R) && \overset{w_Q}{\Rightarrow}\ \textsc{Rel}(E_2, E_1, S)\\
&\textsc{Sub}(L, P) && \tilde{\wedge}\ \textsc{Lbl}(E, L) && \overset{w_Q}{\Rightarrow}\ \textsc{Lbl}(E, P)\\
&\textsc{RSub}(R, S) && \tilde{\wedge}\ \textsc{Rel}(E_1, E_2, R) && \overset{w_Q}{\Rightarrow}\ \textsc{Rel}(E_1, E_2, S)\\
&\textsc{Mut}(L_1, L_2) && \tilde{\wedge}\ \textsc{Lbl}(E, L_1) && \overset{w_Q}{\Rightarrow}\ \tilde{\neg}\textsc{Lbl}(E, L_2)\\
&\textsc{RMut}(R, S) && \tilde{\wedge}\ \textsc{Rel}(E_1, E_2, R) && \overset{w_Q}{\Rightarrow}\ \tilde{\neg}\textsc{Rel}(E_1, E_2, S)
\end{aligned}$$

5.4 Probability Distribution over Uncertain Knowledge Graphs

Combining the logical rules introduced in this section with atoms, such as candidates from the information extraction system (e.g. CandRel(`kyrgyzstan`, `Bishkek`, `hasCapital`)), co-reference information from an entity resolution system (e.g. SameEnt(`kyrgyzstan`, `kyrgz republic`)) and ontological information (e.g. Dom(`hasCapital`, `country`)) we can define a PSL program, $\mathbf{\Pi}$. The inputs to this program instantiate a set of ground rules, R, that consists of the union of groundings from uncertain candidates, $\mathcal{C}$, co-referent entities, $\mathcal{S}$, and ontological relationships, $\mathcal{O}$. The distribution over interpretations, I, generated by PSL corresponds to a probability distribution over knowledge graphs, G:

$$P_{\mathbf{\Pi}}(G) = f(I) = \frac{1}{Z}\exp\left[\sum_{r \in R} w_r \phi_r(I)^p\right]$$

The results of inference provide us with the most likely interpretation, or soft-truth assignments to entities, labels and relations that comprise the knowledge graph. By choosing a threshold on the soft-truth values in the interpretation, we can select a high-precision set of facts to construct a knowledge graph.

6 Experimental Evaluation

6.1 Datasets and Experimental Setup

We evaluate our method on two different datasets: a synthetic knowledge base derived from the LinkedBrainz project [16], which maps data from the MusicBrainz community using ontological information from the MusicOntology [17] as well as web-extraction data from the Never-Ending Language Learning (NELL) project [3]. Our goal is to assess the utility of knowledge graph identification, formulated as a PSL model, at inferring a knowledge graph from noisy data. Additionally, we contrast two very different evaluation settings. In the first, as used in previous work [9] inference is limited to a subset of the knowledge graph generated from the test or

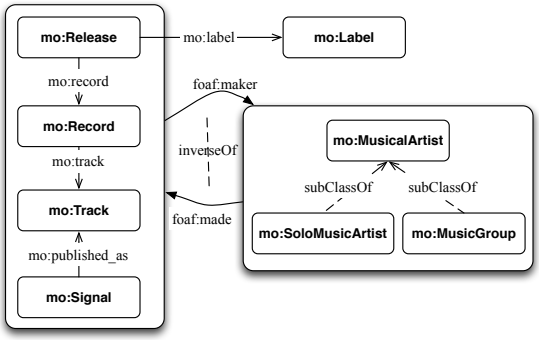


Fig. 2. Subset of Music Ontology mapped using LinkedBrainz for MusicBrainz data in our synthetic dataset

query set. In the second evaluation setting, inference produces a complete knowledge graph, which is not restricted by the test set but employs a soft-truth threshold for atoms. We provide documentation, code and datasets to replicate our results on GitHub[1].

MusicBrainz. MusicBrainz is a community-driven, open-source, structured database for music metadata, including information about artists, albums, and tracks, The Music Ontology is built on top of many well known ontologies, such as FRBR [18] and FOAF [19], and has been used widely, for instance in BBC Music Linked Data sites [20]. However, the relational data available from MusicBrainz are expressed in a proprietary schema that does not map directly to the Music Ontology. To bridge this gap, the LinkedBrainz project publishes an RDF mapping between the freely available MusicBrainz data and the Music Ontology using D2RQ [21]. A summary of the labels and relations we use in our data is show in Figure 2. We use an intuitive mapping of ontological relationships to the PSL predicates, using ontological information from FRBR and FOAF classes used by the Music Ontology. Specifically we convert `rdfs:domain` to Dom, `rdfs:range` to Rng, `rdfs:subClassOf` to Sub, `rdfs:subPropertyOf` to RSub, `owl:inverseOf` to Inv, and `owl:disjointWith` to Mut.

Our synthetic knowledge graph uses a sample of data from the LinkedBrainz mapping of the MusicBrainz project[2] and adds noise to generate a realistic data set. To generate a subset of the LinkedBrainz data, we use snowball sampling from a set of tracks in the MusicBrainz dataset to produce a set of recordings, releases, artists and labels. Next, we introduce noise into this graph by randomly removing known facts and adding inconsistent facts as well as generating random confidence values for these facts. This noise can be interpreted as errors introduced by a MusicBrainz user misspelling artist names, accidentally switching input fields, or omitting information when contributing to the knowledge base.

We model these errors by distorting a percentage of the true input data. For labels, we omit known labels and introduce spurious labels for 25% of the facts

[1] `https://github.com/linqs/KnowledgeGraphIdentification`
[2] `http://linkedbrainz.c4dmpresents.org/content/rdf-dump`

in the input data. When dealing with relations, we focus on the `foaf:maker` and `foaf:made` relations between artists and creative works. We randomly remove one of these pair of relations 25% of the time. Finally, 25% of the time we remove the relationship between a work and its artist, and insert a new relationship between the work and a generated artist, adding a SameEnt for these two artists. The confidence values for facts found in the input are generated from a Normal$(.7, .2)$ distribution while inconsistent facts have lower confidence values generated from a Normal$(.3, .2)$ distribution. The high variance in these distributions ensures a significant overlap. For the SameEnt the similarity values are generated from a Normal$(.9, .1)$ distribution. In all cases, the distribution is thresholded to the $[0, 1]$ range.

We summarize important data statistics in Table 1. In our experiments, we represent the noisy relations and labels of the knowledge graph as candidate facts in PSL with the predicates CandLbl and CandRel. During evaluation, we use the PSL program for knowledge graph identification to infer the most probable knowledge graph. In this setting, we use quadratic combinations of static weights for all rules, where $w_{CL} = w_{CR} = 1$, $w_{EL} = w_{ER} = 25$ and $w_O = 100$. We evaluate our results by comparing to the true knowledge graph used to generate the data, and include false labels corresponding to spurious data we introduce.

NELL. The goal of NELL is to iteratively generate a knowledge base. In each iteration, NELL uses facts learned from the previous iteration and a corpus of web pages to generate a new set of candidate facts. NELL selectively promotes those candidates that have a high confidence from the extractors and obey ontological constraints with the existing knowledge base to build a high-precision knowledge base. We present experimental results on the 165th iteration of NELL, using the candidate facts, promoted facts and ontological relationships that NELL used during that iteration. We summarize the important statistics of this dataset in Table 1. Due to the diversity of the web, the data from NELL is larger, includes more types of relations and categories, and has more ontological relationships than our synthetic data.

NELL uses diverse extraction sources, and in our experiments we use distinct predicates CandLbl_T and CandRel_T for the sources CBL, CMC, CPL, Morph, and SEAL while the remaining sources, which do not contribute a significant number of facts, are represented with CandLbl and CandRel predicates. In addition to candidate facts, NELL uses a heuristic formula to "promote" candidates in each iteration of the system into a knowledge base, however these promotions are often noisy so the system assigns each promotion a confidence value. We represent these promoted candidates from previous iterations as an additional source with corresponding candidate predicates.

In addition to data from NELL, we use data from the YAGO database [22] as part of our entity resolution approach. Our model uses a SameEnt predicate to capture the similarity of two entities. To correct against the multitude of variant spellings found in the data, we use a mapping technique from NELL's entities to Wikipedia articles. We then define a similarity function on the article URLs, using the similarity as the soft-truth value of the SameEnt predicate.

The YAGO database contains entities which correspond to Wikipedia articles, variant spellings and abbreviations of these entities, and associated WordNet categories. Our approach to entity resolution matches entity names in NELL with YAGO entities. We perform selective stemming on the NELL entities, employ blocking on candidate labels, and use a case-insensitive string match to find corresponding YAGO entities. Once we find a matching set of YAGO entities, we can generate a set of Wikipedia URLs that map to the corresponding NELL entities. We can judge the similarity of two entities by computing a set-similarity measure on the Wikipedia URLs associated with the entities. For our similarity score we use the Jaccard index, the ratio of the size of the set intersection and the size of the set union.

In our experiments using NELL, we consider two scenarios. The first is similar to experimental setup in [9] where rule weights are learned using training data and predictions are made on a limited neighborhood of the test set. The neighborhood used in this previous work attempts to improve scalability by generating a grounding of the test set and only including atoms that are not trivially satisfied in this grounding. In practice, this produces a neighborhood that is distorted by omitting atoms that may contradict those in the test set. For example, if ontological relationships such as SUB(`country`,`location`) and MUT(`country`, `city`) are present, the test set atom LBL(`kyrgyzstan`,`country`) would not introduce LBL(`kyrgyzstan`,`city`) or LBL(`country`,`location`) into the neighborhood, even if contradictory data were present in the input candidates. By removing the ability to reason about contradictory information, we believe this evaluation setting diminishes the true difficulty of the problem. We validate our approach on this setting, but also present results from a more realistic setting. In the second scenario we perform inference independently of the test set, lazily generating truth values for atoms supported by the input data, using a soft-truth value threshold of .01. This second setting allows us to infer a complete knowledge graph similar to the MusicBrainz setting.

6.2 Knowledge Graph Identification Results for MusicBrainz

Our experiments on MusicBrainz data attempt to recover the complete knowledge graph despite the addition of noise which introduces uncertainty for facts, removes true information and adds spurious labels and relations. We evaluate a number of variants on their ability to recover this knowledge graph. We measure performance using a number of metrics: the area under the precision-recall curve (AUC), as well as the precision, recall and F1 score at a soft-truth threshold of .5, as well as the maximum F1 score on the dataset. Due to the high variance of confidence values and large number of true facts in the ground truth, the maximum F1 value occurs at a soft-truth threshold of 0, where recall is maximized, in all variants. These results are summarized in Table 2.

The first variant we consider uses only the input data, setting the soft-truth value equal to the generated confidence value as an indicator of the underlying noise in the data. The baseline results use only the candidate rules we introduced

Table 1. Summary of dataset statistics for NELL and MusicBrainz, including (a) the number of candidate facts in input data, the distinct relations and labels present, and (b) the number of ontological relationships defined between these relations and labels

(a)

	NELL	MusicBrainz
Cand. Label	1.2M	320K
Cand. Rel	100K	490K
Promotions	440K	0
Unique Labels	235	19
Unique Rels	221	8

(b)

	NELL	MusicBrainz
Dom	418	8
Rng	418	8
Inv	418	2
Mut	17.4K	8
RMut	48.5K	0
Sub	288	21
RSub	461	2

Table 2. A comparison of knowledge graph identification methods on MusicOntology data shows knowledge graph identification effectively combines the strengths of graph identification and reasoning with ontological information and produces superior results

Method	AUC	Prec	Recall	F1	Max F1
Baseline	0.672	0.946	0.477	0.634	0.788
PSL-EROnly	0.797	0.953	0.558	0.703	0.831
PSL-OntOnly	0.753	0.964	0.605	0.743	0.832
PSL-KGI-Complete	**0.901**	**0.970**	**0.714**	**0.823**	**0.919**

in subsection 5.1. We improve upon this data by adding either the entity resolution rules introduced in subsection 5.2, which we report as PSL-EROnly, or with weighted rules capturing ontological constraints introduced in subsection 5.3. Finally, we combine all the elements of knowledge graph identification introduced in section 5 and report these results as PSL-KGI-Complete. The results on the baseline demonstrate the magnitude of noise in the input data; less than half the facts in the knowledge graph can be correctly inferred. Reasoning jointly about co-referent entities, as in graph identification, improves results. Using ontological constraints, as previous work in improving extraction in this domain has, also improves results as well. Comparing these two improvements, adding entity resolution has a higher AUC, while ontological constraints show a greater improvement in F1 score. However, when these two approaches are combined, as they are in knowledge graph identification, results improve dramatically. Knoweldge graph identification increases AUC, precision, recall and F1 substantially over the the other variants, improving AUC and F1 over 10% compared to the more competitive baseline methods. Overall, we are able to infer 71.4% of true relations while maintaining a precision of .97. Moreover, a high AUC of .901 suggests that knowledge graph identification balances precision and recall for a wide range of parameter values.

6.3 Knowledge Graph Identification Results for NELL

Comparison to Previous Work. While results on data with synthetic noise confirm our hypothesis, we are particularly interested in the results on a large, noisy real-world dataset. We compare our method to data from iteration 165 of NELL using previously reported results on a manually-labeled evaluation set [9]. A summary of these results is shown in Table 3. The first method we compare to is a baseline similar to the one used in the MusicBrainz results where candidates are given a soft-truth value equal to the extractor confidence (averaged across extractors when appropriate). Results are reported at a soft-truth threshold of .45 which maximizes F1.

We also compare the default strategy used by the NELL project to choose candidate facts to include in the knowledge base. Their method uses the ontology to check the consistency of each proposed candidate with previously promoted facts already in the knowledge base. Candidates that do not contradict previous knowledge are ranked using a heuristic rule based on the confidence scores of the extractors that proposed the fact, and the top candidates are chosen for promotion subject to score and rank thresholds. Note that the NELL method includes judgments for all input facts, not just those in the test set.

The third method we compare against is the best-performing MLN model from [9], that expresses ontological constraints, and candidate and promoted facts through logical rules similar to those in our model. The MLN uses additional predicates that have confidence values taken from a logistic regression classifier trained using manually labeled data. The MLN uses hard ontological constraints, learns rule weights considering rules independently and using logistic regression, scales weights by the extractor confidences, and uses MC-Sat with a restricted set of atoms to perform approximate inference, reporting output at a .5 marginal probability cutoff, which maximizes the F1 score. The MLN method only generates predictions for a 2-hop neighborhood generated by conditioning on the values of the query set, as described earlier.

Our method, PSL-KGI, uses PSL with quadratic, weighted rules for ontological constraints, entity resolution, and candidate and promoted facts as well as incorporating a prior. We also incorporate the predicates generated for the MLN method for a more equal comparison. We learn weights for all rules, including the prior, using a voted perceptron learning method. The weight learning method generates a set of target values by running inference and conditioning on the training data, and then chooses weights that maximize the agreement with these targets in absence of training data. Since we represent extractor confidence values as soft-truth values, we do not scale the weights of these rules. Using the learned weights, we perform inference on the same neighborhood defined by the query set that is used by the MLN method. We report these results, using a soft-truth threshold of .55 to maximize F1, as PSL-KGI. As Table 3 shows, knowledge graph identification produces modest improvements in both F1 and AUC.

Analyzing Variations of Knowledge Graph Identification. To better understand the contributions of various components of our model, we explore variants that omit one aspect of the knowledge graph identification model.

Table 3. Comparing against previous work on the NELL dataset, knowledge graph identification using PSL demonstrates a substantive improvement

Method	AUC	Prec	Recall	F1
Baseline	0.873	0.781	0.881	0.828
NELL	0.765	0.801	0.580	0.673
MLN	0.899	**0.837**	0.837	0.836
PSL-KGI	**0.904**	0.777	**0.944**	**0.853**

Table 4. Comparing variants of PSL graph identification show the importance of ontological information, but the best performance is achieved when all of the components of knowledge graph identification are combined

Method	AUC	Prec	Recall	F1
PSL-NoSrcs	0.900	0.770	**0.955**	0.852
PSL-NoER	0.899	0.778	0.944	**0.853**
PSL-NoOnto	0.887	**0.813**	0.839	0.826
PSL-KGI	**0.904**	0.777	0.944	**0.853**

Table 5. Producing a complete knowledge graph reduces performance on the test set, suggesting that the true complexity of the problem is masked when generating a limited set of inferences

Method	AUC	Prec	Recall	F1
NELL	**0.765**	**0.801**	0.580	0.673
PSL-KGI-Complete	0.718	0.709	**0.929**	**0.804**
PSL-KGI	0.904	0.777	0.944	0.853

PSL-NoSrcs removes predicates $\textsc{CandLbl}_T$ and $\textsc{CandRel}_T$ for different candidate sources, replacing them with a single CandLbl or CandRel with the average confidence value across sources. PSL-NoER removes rules from subsection 5.2 used to reason about co-referent entities. PSL-NoOnto removes rules from subsection 5.3 that use ontological relationships to constrain the knowledge graph. While source information and entity resolution both provide benefits, ontological information is clearly a principal contributor to the success of knowledge graph identification. One drawback of our comparisons to previous work is the restriction of the model to a small set of inference targets. The construction of this set obscures some of the challenges presented in real-world data, such as conflicting evidence. To assess the performance of our method in a setting where inference targets do not restrict potentially contradictory inferences, we also ran knowledge graph identification using the same learned weights but with no predefined set of targets, allowing lazy inference to produce a complete knowledge graph. The resulting inference produces a total of 4.9M total facts, which subsumes the test set. We report results on the test set as PSL-KGI-Complete. Allowing the model to optimize on the full knowledge graph instead

of just the test set reduced the performance as measured by the particular test set, suggesting that the noise introduced by conflicting evidence does have a significant impact on results. Compared to the NELL scoring method, KGI has lower AUC and precision but higher recall and F1. One possible explanation for this lackluster performance may be the use of weights learned for a different setting. For example, during weight learning the weights for the MUT rule dropped significantly. However, as results on the MusicBrainz data show, knowledge graph identification can be very powerful at recovering a full knowledge graph.

Scalability. One advantage of using PSL for knowledge graph identification is the ability to frame complex joint reasoning as a convex optimization. Knowledge graph identification implemented in PSL can handle problems from real-world datasets like NELL, which include millions of candidate facts. Inference when an explicit query set of 70K facts is given (PSL-KGI) requires a mere 10 seconds. The MLN method we compare against takes a few minutes to an hour to run for the same setting. When inferring a complete knowledge graph without known query targets, as in the LinkedBrainz and complete NELL experiments, inference with MLNs is infeasible. In contrast, knowledge graph identification on the NELL dataset can produce the complete knowledge graph containing 4.9M facts in only 130 minutes. The ability to produce complete knowledge graphs in these realistic settings is an important feature of our implementation of knowledge graph identification.

7 Conclusion

We have described how to formulate the problem of *knowledge graph identification*: jointly inferring a knowledge graph from the noisy output of an information extraction system through a combined process of determining co-referent entities, predicting relational links, collectively classifying entity labels, and enforcing ontological constraints. Using PSL, we illustrate the benefits of our approach on two knowledge graph inference problems: synthetic data from MusicBrainz and noisy, real-world web extractions from NELL. On both datasets, knowledge graph identification produces superior results by combining the strengths of ontological reasoning with graph identification. Moreover, our method is solved through efficient convex optimization allowing previously infeasible problems to be solved on the order of minutes. In the future, we hope to apply knowledge graph identification to larger, more varied problems with richer ontological relationships.

Acknowledgments. We would like to thank Shangpu Jiang and Daniel Lowd for sharing their data and offering enthusiastic assistance. This work was partially supported by NSF CAREER grant 0746930 and NSF grants IIS1218488 and CCF0937094. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

1. Ji, H., Grishman, R., Dang, H.: Overview of the Knowledge Base Population Track. In: Text Analysis Conference (2011)
2. Artiles, J., Mayfield, J.: Workshop on Knowledge Base Population. In: Artiles, J., Mayfield, J. (eds.) Text Analysis Conference (2012)
3. Carlson, A., Betteridge, J., Kisiel, B., Settles, B., Hruschka, E.R., Mitchell, T.M.: Toward an Architecture for Never-Ending Language Learning. In: AAAI (2010)
4. Etzioni, O., Banko, M., Soderland, S., Weld, D.S.: Open Information Extraction from the Web. Communications of the ACM 51(12) (2008)
5. Pasca, M., Lin, D., Bigham, J., Lifchits, A., Jain, A.: Organizing and Searching the World Wide Web of Facts-Step One: the One-million Fact Extraction Challenge. In: AAAI (2006)
6. Singhal, A.: Introducing the Knowledge Graph: Things, Not Strings, Official Blog, of Google (2012), http://goo.gl/zivFV
7. Broecheler, M., Mihalkova, L., Getoor, L.: Probabilistic Similarity Logic. In: UAI (2010)
8. Cohen, W., McAllester, D., Kautz, H.: Hardening Soft Information Sources. In: KDD (2000)
9. Jiang, S., Lowd, D., Dou, D.: Learning to Refine an Automatically Extracted Knowledge Base Using Markov Logic. In: ICDM (2012)
10. Richardson, M., Domingos, P.: Markov Logic Networks. Machine Learning 62(1-2) (2006)
11. Namata, G.M., Kok, S., Getoor, L.: Collective Graph Identification. In: KDD (2011)
12. Memory, A., Kimmig, A., Bach, S.H., Raschid, L., Getoor, L.: Graph Summarization in Annotated Data Using Probabilistic Soft Logic. In: Workshop on Uncertainty Reasoning for the Semantic Web (URSW) (2012)
13. Yao, L., Riedel, S., McCallum, A.: Collective Cross-Document Relation Extraction Without Labelled Data. In: EMNLP (2010)
14. Kimmig, A., Bach, S.H., Broecheler, M., Huang, B., Getoor, L.: A Short Introduction to Probabilistic Soft Logic. In: NIPS Workshop on Probabilistic Programming (2012)
15. Bach, S.H., Broecheler, M., Getoor, L., O'Leary, D.P.: Scaling MPE Inference for Constrained Continuous Markov Random Fields with Consensus Optimization. In: NIPS (2012)
16. Dixon, S., Jacobson, K.: LinkedBrainz - A project to provide MusicBrainz NGS as Linked Data, http://linkedbrainz.c4dmpresents.org/
17. Raimond, Y., Abdallah, S., Sandler, M.: The Music Ontology. In: International Conference on Music Information Retrieval (2007)
18. Davis, I., Newman, R., Darcus, B.: Expression of Core FRBR Concepts in RDF (2005), http://vocab.org/frbr/core.html
19. Brickley, D., Miller, L.: FOAF Vocabulary Specification 0.98 (2010), http://xmlns.com/foaf/spec/20100809.html
20. Kobilarov, G., Scott, T., Raimond, Y., Oliver, S., Sizemore, C., Smethurst, M., Bizer, C., Lee, R.: Media Meets Semantic Web–How The BBC uses DBpedia and Linked Data to Make Connections. In: Aroyo, L., Traverso, P., Ciravegna, F., Cimiano, P., Heath, T., Hyvönen, E., Mizoguchi, R., Oren, E., Sabou, M., Simperl, E. (eds.) ESWC 2009. LNCS, vol. 5554, pp. 723–737. Springer, Heidelberg (2009)
21. Bizer, C., Seaborne, A.: D2RQ–Treating Non-RDF Databases as Virtual RDF Graphs. In: ISWC (2004)
22. Suchanek, F.M., Kasneci, G., Weikum, G.: Yago: A Core of Semantic Knowledge. In: WWW (2007)

Ontology-Based Data Access: *Ontop* of Databases

Mariano Rodríguez-Muro[1], Roman Kontchakov[2], and Michael Zakharyaschev[2]

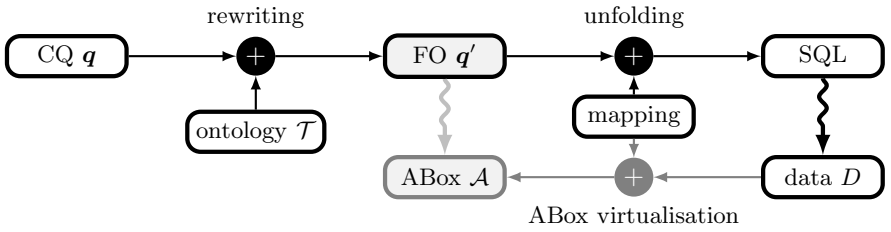
[1] Faculty of Computer Science, Free University of Bozen-Bolzano, Italy
[2] Department of Computer Science and Information Systems, Birkbeck, University of London, U.K.

Abstract. We present the architecture and technologies underpinning the OBDA system *Ontop* and taking full advantage of storing data in relational databases. We discuss the theoretical foundations of *Ontop*: the tree-witness query rewriting, $\mathcal{T}$-mappings and optimisations based on database integrity constraints and SQL features. We analyse the performance of *Ontop* in a series of experiments and demonstrate that, for standard ontologies, queries and data stored in relational databases, *Ontop* is fast, efficient and produces SQL rewritings of high quality.

1 Introduction

Ontology-based data access (OBDA) [6,11,22] is regarded as a key ingredient for the new generation of information systems, especially for Semantic Web applications that involve large amounts of data. In the OBDA paradigm, an ontology defines a high-level global schema and provides a vocabulary for user queries, thus isolating the user from the details of the structure of data sources (which can be relational databases, triple stores, datalog engines, etc.). The OBDA system transforms user queries into the vocabulary of the data and then delegates the actual query evaluation to the data sources.

In this paper, we concentrate on OBDA with ontologies given in OWL 2 QL, a profile of OWL 2 designed to support rewriting of conjunctive queries (CQs) over ontologies into first-order (FO) queries. A standard architecture of such an OBDA system over relational data sources can be represented as follows:



The user is given an OWL 2 QL ontology $\mathcal{T}$ and can formulate CQs $\boldsymbol{q}(\boldsymbol{x})$ in the signature of $\mathcal{T}$. The system rewrites $\boldsymbol{q}$ and $\mathcal{T}$ into an FO-query $\boldsymbol{q}'(\boldsymbol{x})$, called a *rewriting* of $\boldsymbol{q}$ and $\mathcal{T}$, such that $(\mathcal{T},\mathcal{A}) \models \boldsymbol{q}(\boldsymbol{a})$ iff $\mathcal{A} \models \boldsymbol{q}'(\boldsymbol{a})$, for any set $\mathcal{A}$ of ground atoms (called an ABox) in the signature of $\mathcal{T}$ and any tuple $\boldsymbol{a}$

H. Alani et al. (Eds.): ISWC 2013, Part I, LNCS 8218, pp. 558–573, 2013.

of individuals in $\mathcal{A}$. A number of different rewriting techniques have been proposed and implemented for OWL 2 QL (PerfectRef [22], Presto/Prexto [27,26], Rapid [5], the tree-witness rewriting [15]) and its extensions ([16], Nyaya [9], Requiem/Blackout [20,21], Clipper [7]).

The rewriting $\boldsymbol{q}'$ is formulated in the signature of $\mathcal{T}$ and, before evaluation, has to be further transformed into the vocabulary of the data source D. For instance, $\boldsymbol{q}'$ can be *unfolded* into an SQL query by means of a GAV mapping $\mathcal{M}$ relating the signature of $\mathcal{T}$ to the vocabulary of D. Strangely enough, mappings and unfoldings have largely been ignored by query rewriting algorithms (with Mastro-I [22] being an exception), partly because the data was assumed to be given as an ABox (say, as a universal table in a database or as a triple store). We consider the query transformation process as consisting of two steps—query rewriting and unfolding—and argue that this brings practical benefits (even in the case of seemingly trivial mappings for universal tables or triple stores).

The performance of first OBDA systems based on the architecture above was marred by large rewritings that could not be processed by RDBMSs, which led the OBDA community to intensive investigations of rewriting techniques and optimisations. There are 3 main reasons for large CQ rewritings and unfoldings:

(E) Sub-queries of $\boldsymbol{q}$ with existentially quantified variables can be folded in many different ways to match the canonical models of possible knowledge bases $(\mathcal{T}, \mathcal{A})$, all of which must be reflected in the rewriting $\boldsymbol{q}'$.

(H) Classes/properties occurring in $\boldsymbol{q}$ can have many subclasses/subproperties according to $\mathcal{T}$, which all have to be included in the rewriting $\boldsymbol{q}'$.

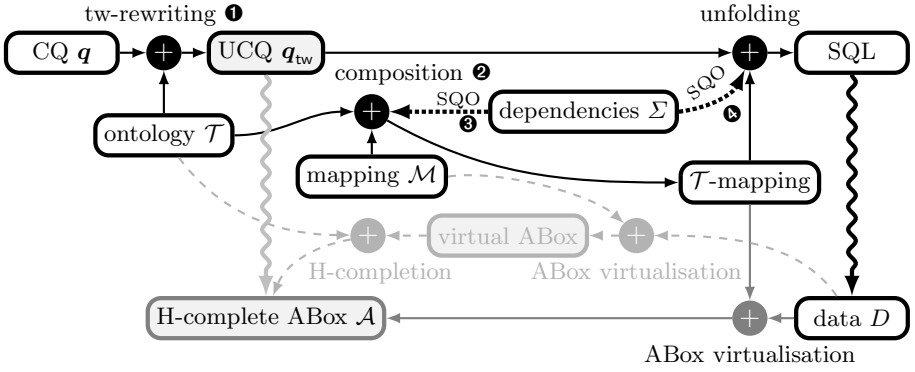
(M) The mapping $\mathcal{M}$ can have multiple definitions of the ontology terms, which may result in an exponential blowup when $\boldsymbol{q}'$ is unfolded into a (most suitable for RDBMSs) union of Select-Project-Join queries.

In fact, most of the proposed techniques produce rewritings in the form of unions of CQs (UCQs) and try to tame (E) using various optimisations in unification strategies to reduce the size of rewritings, with expensive CQ containment as the last resort. Presto [27] and the tree-witness rewriting [15] use nonrecursive datalog to deal with (H); this, however, is of little help if a further transformation to a UCQ is required. The combined approach [17] constructs finite representations of (in general) infinite canonical models of $(\mathcal{T}, \mathcal{A})$ thereby totally removing (H). It also solves (E) for ontologies without role inclusions; otherwise, rewritings can still be of exponential size, or the filtering procedure [19] may have to run exponentially many times.

In theory, (E) turns out to be incurable under the architecture above: there are CQs and OWL 2 QL ontologies for which any FO- (or nonrecursive datalog) rewriting is superpolynomial (or exponential) [13,14], which happens independently of the contribution of (H) and (M); the polynomial rewriting of [10] hides this blowup behind extra existential quantifiers. Fortunately, it seems that only (artificially) complex CQs and ontologies trigger issues with (E). Our experiments show that, for standard benchmark CQs and ontologies, the number of foldings in (E) is small and can be efficiently dealt with by the tree-witness rewriting.

In this paper, we attack both (H) and (M) at *the same time* using two key observations. First, the schema and integrity constraints (dependencies), Σ, of the data source D together with the mapping $\mathcal{M}$ often provide valuable information about the class of possible ABoxes over which the user CQ is rewritten. (These ABoxes are *virtual* representations of D and are not materialised.) For example, if we know that all our virtual ABoxes $\mathcal{A}$ are $\exists$-*complete* with respect to $\mathcal{T}$ (that is, contain witnesses for all $\exists R$ in $\mathcal{T}$) then we can ignore (E); if all $\mathcal{A}$ are *H-complete* (that is, $\mathcal{A}$ contains $A(a)$ whenever it contains $B(a)$ and $\mathcal{T} \models B \sqsubseteq A$, and similarly for properties) then the problem (H) does not exist. Second, we can make the virtual ABoxes H-complete by taking the composition of $\mathcal{T}$ and $\mathcal{M}$ as a new mapping. This composition, called a $\mathcal{T}$-*mapping* [24], can be simplified with the help of Σ and the features of the target query language before being used in the unfolding. As the simplifications use Σ, they preserve correct answers only over database instances satisfying Σ. (Even if the mappings are trivial and the data comes from a universal table or a triple store, it often has a certain structure and satisfies certain constraints, which could be taken into account to make query answering more efficient [12]).

These observations underpin the system *Ontop* (`ontop.inf.unibz.it`) implemented at the Free University of Bozen-Bolzano and available as a plugin for Protégé 4, SPARQL end-point and OWLAPI and Sesame libraries. The process of query rewriting and unfolding in *Ontop* with all optimisations is shown below (the dashed lines show processes that aid explanations but do *not* take place):



This architecture, which is our main theoretical contribution, will be discussed in detail in Section 2. Here we only emphasise the key ingredients:

❶ the tree-witness rewriting $\boldsymbol{q}_{\text{tw}}$ assumes the virtual ABoxes to be H-complete; it separates the topology of $\boldsymbol{q}$ from the taxonomy defined by $\mathcal{T}$, is fast in practice and produces short UCQs;

❷ the $\mathcal{T}$-mapping combines the system mapping $\mathcal{M}$ with the taxonomy of $\mathcal{T}$ to ensure H-completeness of virtual ABoxes;

❸ the $\mathcal{T}$-mapping is simplified using the Semantic Query Optimisation (SQO) technique and SQL features; the $\mathcal{T}$-mapping is constructed and optimised for the given $\mathcal{T}$ and Σ only once, and is used to unfold all rewritings $\boldsymbol{q}_{\text{tw}}$;

❹ the unfolding algorithm uses SQO to produce small and efficient SQL queries.

In Section 3, we evaluate the performance of *Ontop* and compare it with other systems using a number of standard ontologies, including LUBM with generated data and the Movie Ontology with real data. Our experimental results show that UCQ rewritings over arbitrary ABoxes are not scalable in the presence of class and property hierarchies; in contrast to that, rewritings of real-world queries and ontologies over H-complete ABoxes (or equivalent datalog rewritings) turn out to be unions of few (at most two, in our experiments) CQs whose size does not exceed (in fact, is often smaller than) the size of the original query. Class and property hierarchies can be tackled by optimisations of $\mathcal{T}$-mappings and the SQO, which use the structure of databases and integrity constraints, so that *Ontop* automatically produces SQL queries of reasonably high quality. As a result, *Ontop* successfully competes with and often outperforms systems based on materialisation of inferences.

2 The Architecture of *Ontop*

We begin by describing the three main ingredients of *Ontop*: the tree-witness rewriting over H-complete ABoxes, $\mathcal{T}$-mappings and the unfolding algorithm. To avoid long formulas, we use the DL parlance [2] for OWL 2 QL ontologies and the datalog notation for conjunctive queries. Thus, subclass axioms are of the form $A_1 \sqsubseteq A_2$, for concept (class) names A_i; property inclusions are $R_1 \sqsubseteq R_2$, where the R_i are role (object and datatype property) names or their inverses; and property P domain and range axioms are $\exists P \sqsubseteq A_1$ and $\exists P^- \sqsubseteq A_2$, respectively. Conjunctive queries (CQs) are of the form $\boldsymbol{q}(\boldsymbol{x}) \leftarrow \alpha_1, \dots, \alpha_n$, where $\boldsymbol{x}$ is a vector of answer variables and each α_i is a unary or binary atom (the variables in the α_i that are not in $\boldsymbol{x}$ are *existentially quantified*). Throughout the paper, we identify atoms $P^-(y, x)$ and $P(x, y)$ (in query heads, bodies and ABoxes).

Suppose we are given a CQ $\boldsymbol{q}(\boldsymbol{x})$ and an OWL 2 QL ontology $\mathcal{T}$. *Ontop* starts its work by constructing the *semantic-based* tree-witness rewriting of $\boldsymbol{q}$ and $\mathcal{T}$ over H-complete ABoxes. We say that an ABox $\mathcal{A}$ is *H-complete with respect to* $\mathcal{T}$ in case it satisfies the following conditions:

$$A(a) \in \mathcal{A} \quad \text{if} \quad A'(a) \in \mathcal{A}, \mathcal{T} \models A' \sqsubseteq A \quad \text{or} \quad R(a, b) \in \mathcal{A}, \mathcal{T} \models \exists R \sqsubseteq A,$$
$$P(a, b) \in \mathcal{A} \quad \text{if} \quad R(a, b) \in \mathcal{A} \text{ and } \mathcal{T} \models R \sqsubseteq P.$$

2.1 Tree-Witness Rewriting over H-Complete ABoxes

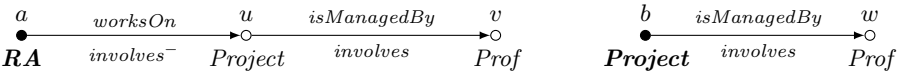
We explain the essence of the tree-witness rewriting using an example; a formal definition can be found in [25]. Consider an ontology $\mathcal{T}$ with the axioms

$$RA \sqsubseteq \exists worksOn.Project, \qquad Project \sqsubseteq \exists isManagedBy.Prof, \tag{1}$$
$$worksOn^- \sqsubseteq involves, \qquad isManagedBy \sqsubseteq involves, \tag{2}$$

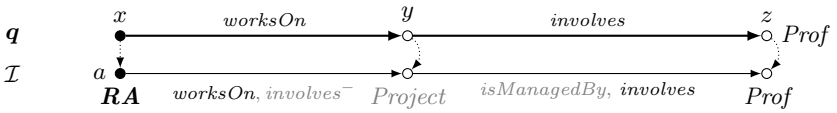
and the CQ $\boldsymbol{q}(\boldsymbol{x})$ asking to find those who work with professors:

$$\boldsymbol{q}(x) \leftarrow worksOn(x, y),\ involves(y, z),\ Prof(z).$$

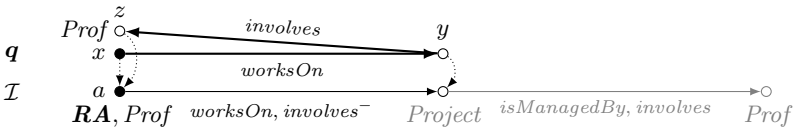
Observe that if a model $\mathcal{I}$ of $(\mathcal{T}, \mathcal{A})$, for some ABox $\mathcal{A}$, contains individuals $a \in RA^{\mathcal{I}}$ and $b \in Project^{\mathcal{I}}$ then $\mathcal{I}$ must also contain the following fragments:



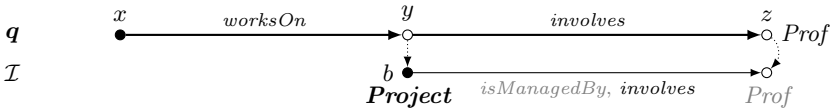
where the points u, v, w are not necessarily named individuals from the ABox, but can be (anonymous) witness for the existential quantifiers of (1) (or labelled nulls in the chase); we say that these fragments are *generated* by RA and $Project$, respectively, and use the bold-faced font to indicate that. It follows that a is an answer to $\boldsymbol{q}(x)$ whenever a is an instance of $RA^{\mathcal{I}}$, in which case the atoms of $\boldsymbol{q}$ (thick lines) are mapped to the fragment generated by RA as follows:



Alternatively, we obtain the following match (provided that a is also in $Prof^{\mathcal{I}}$):



Another option is to map x and y to ABox individuals, a and b, and if b is in $Project^{\mathcal{I}}$, then the last two atoms of $\boldsymbol{q}$ can be mapped to the anonymous part generated by $Project$:



Finally, all the atoms of $\boldsymbol{q}$ can be mapped to ABox individuals. The possible ways of mapping parts of the CQ to the anonymous part of the models are called *tree witnesses*. The tree-witnesses for $\boldsymbol{q}$ found above give the following UCQ *tree-witness rewriting* $\boldsymbol{q}_{\mathsf{tw}}(x)$ of $\boldsymbol{q}(x)$ and $\mathcal{T}$ over H-complete ABoxes:

$$
\begin{aligned}
&\boldsymbol{q}_{\mathsf{tw}}(x) \leftarrow \boldsymbol{RA}(x),\\
&\boldsymbol{q}_{\mathsf{tw}}(x) \leftarrow \mathit{Prof}(x), \boldsymbol{RA}(x),\\
&\boldsymbol{q}_{\mathsf{tw}}(x) \leftarrow \mathit{worksOn}(x, y), \boldsymbol{Project}(y),\\
&\boldsymbol{q}_{\mathsf{tw}}(x) \leftarrow \mathit{worksOn}(x, y), \mathit{involves}(y, z), \mathit{Prof}(z).
\end{aligned}
$$

(It is to be noted that $\boldsymbol{q}_{\mathsf{tw}}(x)$ is *not* a rewriting of $\boldsymbol{q}(x)$ and $\mathcal{T}$ over *all* ABoxes.)

Having computed the UCQ $\boldsymbol{q}_{\mathsf{tw}}$, *Ontop* simplifies it using two optimisations. First, it applies a subsumption algorithm to remove redundant CQs from the union: for example, the first query in the example above subsumes the second, which can be safely removed. It also reduces the size of the individual CQs in

the union using the following observation: any CQ $\boldsymbol{q}$ (viewed as a set of atoms) has the same certain answers over *H-complete* ABoxes as

$$\boldsymbol{q} \setminus \{A(x)\}, \qquad \text{if } A'(x) \in \boldsymbol{q} \text{ and } \mathcal{T} \models A' \sqsubseteq A \text{ with } A' \neq A, \tag{3}$$

$$\boldsymbol{q} \setminus \{A(x)\}, \qquad \text{if } R(x,y) \in \boldsymbol{q} \text{ and } \mathcal{T} \models \exists R \sqsubseteq A, \tag{4}$$

$$\boldsymbol{q} \setminus \{P(x,y)\}, \qquad \text{if } R(x,y) \in \boldsymbol{q} \text{ and } \mathcal{T} \models R \sqsubseteq P \text{ with } R \neq P, \tag{5}$$

Surprisingly, such a simple optimisation, especially (4) for domains and ranges, makes rewritings substantially shorter [27,9].

We have to bear in mind, however, that in theory, the size of the resulting UCQ rewritings can be very large: there exists [13,14] a sequence of $\boldsymbol{q}_n$ and $\mathcal{T}_n$ generating exponentially many (in $|\boldsymbol{q}_n|$) tree witnesses, and *any* first-order (or nonrecursive datalog) rewriting of $\boldsymbol{q}_n$ and $\mathcal{T}_n$ is of superpolynomial (or exponential) size (unless it employs $|\boldsymbol{q}_n|$-many additional existentially quantified variables [10]). On the other hand, to generate many tree witnesses, the CQ $\boldsymbol{q}$ must have many subqueries that can be matched in the canonical models, which requires both $\boldsymbol{q}$ and $\mathcal{T}$ to be quite sophisticated, with $\boldsymbol{q}$ 'mimicking' parts of the canonical models for $\mathcal{T}$. To the best of our knowledge, this never happens in real-world CQs and ontologies used for OBDA. More often than not, they do not generate tree witnesses at all; see Section 3.1. It is also known [15, Theorem 21] that, if the query and ontology do not contain fragments as in the example considered above, then the number of tree witnesses is polynomial.

2.2 Optimising $\mathcal{T}$-Mappings

In a typical scenario for *Ontop*, the data comes from a relational database rather than an ABox. A database schema [1] contains predicate symbols (with their arity) for both stored database relations and views (with their definitions in terms of stored relations) as well as a set Σ of integrity constraints (in the form of inclusion and functional dependencies). Any instance $\boldsymbol{I}$ of the database schema must satisfy its integrity constraints Σ. The vocabularies of a database schema and an ontology are linked together by means of mappings. We define a *mapping*, $\mathcal{M}$, as a set of GAV rules of the form

$$S(\boldsymbol{x}) \leftarrow \varphi(\boldsymbol{x}, \boldsymbol{z}),$$

where S is a class or property name in the ontology and $\varphi(\boldsymbol{x}, \boldsymbol{z})$ a conjunction of atoms with database relations (both stored relations and views) and a *filter*, that is, a Boolean combination of built-in predicates such as $=$ and $<$. (Note that, by including views in the schema, we can express any SQL query in mappings.) Given a mapping $\mathcal{M}$ and a data instance $\boldsymbol{I}$, the ground atoms

$$S(\boldsymbol{a}), \quad \text{for } S(\boldsymbol{x}) \leftarrow \varphi(\boldsymbol{x}, \boldsymbol{z}) \text{ in } \mathcal{M} \text{ and } \boldsymbol{I} \models \exists \boldsymbol{z}\, \varphi(\boldsymbol{a}, \boldsymbol{z}),$$

comprise the ABox, $\mathcal{A}_{\boldsymbol{I},\mathcal{M}}$, which is called the *virtual ABox* for $\mathcal{M}$ over $\boldsymbol{I}$. We can now define *certain answers* to a CQ $\boldsymbol{q}$ over an ontology $\mathcal{T}$ linked by a mapping $\mathcal{M}$ to a database instance $\boldsymbol{I}$ as certain answers to $\boldsymbol{q}$ over $(\mathcal{T}, \mathcal{A}_{\boldsymbol{I},\mathcal{M}})$.

The tree-witness rewriting $\boldsymbol{q}_{\mathsf{tw}}$ of $\boldsymbol{q}$ and $\mathcal{T}$ works only for H-complete ABoxes. An obvious way to define such ABoxes is to take the composition $\mathcal{M}^{\mathcal{T}}$ of $\mathcal{M}$ and the inclusions in $\mathcal{T}$ given by

$$\begin{aligned} A(x) &\leftarrow \varphi(x, \boldsymbol{z}) && \text{if } A'(x) \leftarrow \varphi(x, \boldsymbol{z}) \in \mathcal{M} \text{ and } \mathcal{T} \models A' \sqsubseteq A, \\ A(x) &\leftarrow \varphi(x, y, \boldsymbol{z}) && \text{if } R(x, y) \leftarrow \varphi(x, y, \boldsymbol{z}) \in \mathcal{M} \text{ and } \mathcal{T} \models \exists R \sqsubseteq A, \\ P(x, y) &\leftarrow \varphi(x, y, \boldsymbol{z}), && \text{if } R(x, y) \leftarrow \varphi(x, y, \boldsymbol{z}) \in \mathcal{M} \text{ and } \mathcal{T} \models R \sqsubseteq P \end{aligned}$$

(we do not distinguish between $P^-(y, x)$ and $P(x, y)$). Thus, to compute answers to $\boldsymbol{q}$ over $\mathcal{T}$ with $\mathcal{M}$ and a database instance $\boldsymbol{I}$, it suffices to evaluate the rewriting $\boldsymbol{q}_{\mathsf{tw}}$ over $\mathcal{A}_{\boldsymbol{I},\mathcal{M}^{\mathcal{T}}}$: for any $\boldsymbol{I}$ and any tuple $\boldsymbol{a}$ of individuals in $\mathcal{A}_{\boldsymbol{I},\mathcal{M}}$,

$$(\mathcal{T}, \mathcal{A}_{\boldsymbol{I},\mathcal{M}}) \models \boldsymbol{q}(\boldsymbol{a}) \quad \text{iff} \quad \mathcal{A}_{\boldsymbol{I},\mathcal{M}^{\mathcal{T}}} \models \boldsymbol{q}_{\mathsf{tw}}(\boldsymbol{a}). \tag{6}$$

Given a CQ $\boldsymbol{q}$ and an ontology $\mathcal{T}$, most OBDA systems first construct a rewriting of $\boldsymbol{q}$ and $\mathcal{T}$ over *arbitrary* ABoxes and then unfold it, using a mapping $\mathcal{M}$, into a union of Select-Project-Join (SPJ) queries, which is forwarded for execution to an RDBMS. By (6), the same result can be achieved by unfolding a rewriting over H-complete ABoxes with the help of the composition $\mathcal{M}^{\mathcal{T}}$. In principle, this may bring some benefits if the SQL query is represented as a union of SPJ queries over views for class and property names, but only if the RDBMS can evaluate such queries efficiently (each view is a union of simple queries, for rules in $\mathcal{M}^{\mathcal{T}}$ listing subclasses and subproperties). On the other hand, there will be no benefit if the query is unfolded into a union of SPJ queries either by the RDBMS or by the OBDA system itself. However, the resulting query will produce duplicating answers if the ontology axioms express the same properties of the application domain as the integrity constraints of the database [23].

For this reason, before applying $\mathcal{M}^{\mathcal{T}}$ to unfold the tree-witness rewriting in *Ontop*, we optimise the mapping using the database integrity constraints Σ. This allows us to (*a*) reduce redundancy in answers, and (*b*) substantially shorten the SQL queries. We say that a mapping $\mathcal{M}$ is a *$\mathcal{T}$-mapping over Σ* if the ABox $\mathcal{A}_{\boldsymbol{I},\mathcal{M}}$ is H-complete with respect to $\mathcal{T}$, for any data instance $\boldsymbol{I}$ satisfying Σ. (The composition $\mathcal{M}^{\mathcal{T}}$ is trivially a $\mathcal{T}$-mapping over any Σ.)

To illustrate the optimisations, we take a simplified IMDb (`www.imdb.com/interfaces`) whose schema contains relations $\mathit{title}[m, t, y]$ with information about movies (ID, title, production year), and $\mathit{castinfo}[p, m, r]$ with information about movie casts (person ID, movie ID, person role), and an ontology MO (`www.movieontology.org`) describing the application domain in terms of, for example, classes *mo:Movie* and *mo:Person*, and properties *mo:cast* and *mo:year*:

$$\begin{aligned} \mathit{mo{:}Movie} &\equiv \exists \mathit{mo{:}title}, & \mathit{mo{:}Movie} &\sqsubseteq \exists \mathit{mo{:}year}, \\ \mathit{mo{:}Movie} &\equiv \exists \mathit{mo{:}cast}, & \exists \mathit{mo{:}cast}^- &\sqsubseteq \mathit{mo{:}Person}. \end{aligned}$$

A mapping $\mathcal{M}$ that relates the ontology terms to the database schema contains, for example, the following rules:

$$\begin{aligned} \mathit{mo{:}Movie}(m), \mathit{mo{:}title}(m, t), \mathit{mo{:}year}(m, y) &\leftarrow \mathit{title}(m, t, y), \\ \mathit{mo{:}cast}(m, p), \mathit{mo{:}Person}(p) &\leftarrow \mathit{castinfo}(p, m, r). \end{aligned}$$

Inclusion Dependencies. Suppose $\mathcal{M} \cup \{S(\boldsymbol{x}) \leftarrow \psi_1(\boldsymbol{x}, \boldsymbol{z})\}$ is a $\mathcal{T}$-mapping over Σ. If there is a more specific rule than $S(\boldsymbol{x}) \leftarrow \psi_1(\boldsymbol{x}, \boldsymbol{z})$ in $\mathcal{M}$, then $\mathcal{M}$ itself is also a $\mathcal{T}$-mapping. To discover such 'more specific' rules, we run the standard query containment check (see, e.g., [1]), but taking account of the inclusion dependencies. For example, since $\mathcal{T} \models \exists \mathit{mo{:}cast} \sqsubseteq \mathit{mo{:}Movie}$, the composition $\mathcal{M}^{\mathrm{MO}}$ of mapping $\mathcal{M}$ and MO contains the following rules for *mo:Movie*:

$$\begin{aligned} \mathit{mo{:}Movie}(m) &\leftarrow \mathit{title}(m, t, y), \\ \mathit{mo{:}Movie}(m) &\leftarrow \mathit{castinfo}(p, m, r). \end{aligned}$$

The latter is redundant as IMDb contains the foreign key (inclusion dependency)

$$\forall m \left(\exists p, r\, \mathit{castinfo}(p, m, r) \rightarrow \exists t, y\, \mathit{title}(m, t, y)\right).$$

Disjunctions in SQL. Another way to reduce the size of a $\mathcal{T}$-mapping is to identify pairs of rules whose bodies are equivalent up to *filters w.r.t. constant values*. This optimisation deals with the rules introduced due to the so-called type (discriminating) attributes [8] in database schemas. For example, the mapping $\mathcal{M}$ for IMDb and MO contains six rules for subclasses of *mo:Person*:

$$\begin{aligned} \mathit{mo{:}Actor}(p) &\leftarrow \mathit{castinfo}(c, p, m, r), (r = 1), \\ &\cdots \\ \mathit{mo{:}Editor}(p) &\leftarrow \mathit{castinfo}(c, p, m, r), (r = 6). \end{aligned}$$

Then the composition $\mathcal{M}^{\mathrm{MO}}$ contains six rules for *mo:Person* that differ only in the last condition $(r = k)$, $1 \leq k \leq 6$. These can be reduced to a single rule:

$$\mathit{mo{:}Person}(p) \leftarrow \mathit{castinfo}(c, p, m, r), (r = 1) \vee \cdots \vee (r = 6).$$

Note that such disjunctions lend themselves to efficient evaluation by RDBMSs.

Materialised ABoxes and Semantic Index. In addition to working with proper relational data sources, *Ontop* supports ABox storage in the form of structureless *universal tables*: a binary relation $\mathit{CA}[\mathit{id}, \mathit{class\text{-}id}]$ and a ternary relation $\mathit{RA}[\mathit{id}_1, \mathit{id}_2, \mathit{property\text{-}id}]$ represent class and property membership assertions. The universal tables give rise to trivial mappings, and *Ontop* implements a technique, the *semantic index* [24], that takes advantage of SQL features in $\mathcal{T}$-mappings for this scenario. The key observation is that since the IDs in the universal tables *CA* and *RA* can be chosen by the system, each class and property name in the TBox $\mathcal{T}$ can be assigned a numeric *index* and a set of numeric *intervals* in such a way that the resulting $\mathcal{T}$-mapping contains simple SQL queries with interval filter conditions. For example, in IMDb, we have

$$\mathit{mo{:}Actor} \sqsubseteq \mathit{mo{:}Artist}, \quad \mathit{mo{:}Artist} \sqsubseteq \mathit{mo{:}Person}, \quad \mathit{mo{:}Director} \sqsubseteq \mathit{mo{:}Person};$$

so we can choose index 1 and interval [1,1] for *mo:Actor*, 2 and [1,2] for *mo:Artist*, 3 and [3,3] for *mo:Director* and 6 and [1,6] for *mo:Person*. This will generate a $\mathcal{T}$-mapping with, for instance,

$$\textit{mo:Person}(p) \leftarrow \textit{CA}(p, \textit{class-id}), (1 \leq \textit{class-id} \leq 6),$$
$$\textit{mo:Artist}(p) \leftarrow \textit{CA}(p, \textit{class-id}), (1 \leq \textit{class-id} \leq 2).$$

So, by choosing appropriate class and property IDs, we effectively construct H-complete ABoxes *without* the expensive forward chaining procedure (and the need to store large amounts of derived assertions). On the other hand, the semantic index $\mathcal{T}$-mappings are based on range expressions that can be evaluated efficiently by RDBMSs using standard B-tree indexes [8].

2.3 Unfolding with Semantic Query Optimisation (SQO)

The unfolding procedure [22] applies SLD-resolution to $\boldsymbol{q}_{\mathsf{tw}}$ and the $\mathcal{T}$-mapping, and returns those rules whose bodies contain only database atoms (cf. partial evaluation [18]). *Ontop* applies SQO [4] to rules obtained at the intermediate steps of unfolding. In particular, it eliminates redundant self-JOIN operations caused by reification of database relations by means of classes and properties. Consider, for example, the CQ

$$\boldsymbol{q}(t, y) \leftarrow \textit{mo:Movie}(m), \textit{mo:title}(m, t), \textit{mo:year}(m, y), (y > 2010).$$

It has no tree witnesses, and so $\boldsymbol{q}_{\mathsf{tw}} = \boldsymbol{q}$. By straightforwardly applying the unfolding to $\boldsymbol{q}_{\mathsf{tw}}$ and the $\mathcal{T}$-mapping $\mathcal{M}$ above, we obtain the query

$$\boldsymbol{q}'_{\mathsf{tw}}(t, y) \leftarrow \textit{title}(m, t_0, y_0), \textit{title}(m, t, y_1), \textit{title}(m, t_2, y), (y > 2010),$$

which requires two (potentially) expensive JOIN operations. However, by using the primary key m of *title*:

$$\forall m\, \forall t_1 \forall t_2 \left(\exists y\, \textit{title}(m, t_1, y) \land \exists y\, \textit{title}(m, t_2, y) \rightarrow (t_1 = t_2)\right),$$
$$\forall m\, \forall y_1 \forall y_2 \left(\exists t\, \textit{title}(m, t, y_1) \land \exists t\, \textit{title}(m, t, y_2) \rightarrow (y_1 = y_2)\right)$$

(a functional dependency with determinant m), we reduce two JOIN operations in the first three atoms of $\boldsymbol{q}'_{\mathsf{tw}}$ to a single atom $\textit{title}(m, t, y)$:

$$\boldsymbol{q}''_{\mathsf{tw}}(t, y) \leftarrow \textit{title}(m, t, y), (y > 2010).$$

Note that these two JOIN operations were introduced to reconstruct the ternary relation from its reification by means of the roles *mo:title* and *mo:year*.

The role of SQO in OBDA systems appears to be much more prominent than in conventional RDBMSs, where it was initially proposed to optimise SQL queries. While some of the SQO techniques reached industrial RDBMSs, it never had a strong impact on the database community because it is costly compared to statistics- and heuristics-based methods, and because most SQL queries are written by highly-skilled experts (and so are nearly optimal anyway). In OBDA scenarios, in contrast, SQL queries are generated automatically, and so SQO becomes the only tool to avoid redundant and expensive JOIN operations [28].

Table 1. Tree-witness UCQ rewritings over H-complete ABoxes

A & S	a_1	a_2	a_3	a_4	a_5	s_1	s_2	s_3	s_4	s_5
tree witnesses	1	1	0	1	0	0	0	0	0	0
CQs in $\boldsymbol{q}_{\mathsf{tw}}$	**2**	**2**	**1**	**2**	**1**	**1**	**1**	**1**	**1**	**1**
atoms in $\boldsymbol{q}$	2	3	5	3	5	1	3	5	5	7
atoms in $\boldsymbol{q}_{\mathsf{tw}}$	2+2	1+3	5	2+3	5	1	1	3	2	4

$\text{LUBM}^{\exists}_{20}$	r_1	r_2	r_3	r_4	r_5	q_1	q_2	q_3	q_4	q_5	q_6	q_7	q_8	q_9
tree witnesses	0	0	0	0	0	1	1	0	1	0	0	0	3	1
CQs in $\boldsymbol{q}_{\mathsf{tw}}$	**1**	**1**	**1**	**1**	**1**	**2**	**2**	**1**	**2**	**1**	**1**	**1**	**1**	**1**
atoms in $\boldsymbol{q}$	2	3	6	3	4	8	4	6	8	5	8	13	13	34
atoms in $\boldsymbol{q}_{\mathsf{tw}}$	2	1	4	1	2	4+6	3+4	5	5+8	4	6	12	6	33

3 Experiments

In this section, we present the results of experiments conducted to evaluate the performance of *Ontop* in comparison with other systems (for details see `sites.google.com/site/ontopiswc13`). We begin by testing the tree-witness rewriter.

3.1 Tree Witnesses: The Topology of *Ontop* Rewritings

We ran the *Ontop* tree-witness rewriter on the usual set of ontologies and CQs: Adolena (A) and StockExchange (S) [20] with the original queries a_1–a_5 and s_1–s_5, respectively, and $\text{LUBM}^{\exists}_{20}$ [19] with queries r_1–r_5 from the Requiem evaluation [20], q_1–q_6 from the combined approach evaluation [19], and q_7–q_9 from the Clipper evaluation [30]. Our aim was to understand the size of the *topological* part of the rewritings that reflects matches into the anonymous part of the canonical models (as opposed to the taxonomical one). Table 1 shows the number of tree witnesses, the number of CQs in the rewriting, and the number of atoms in the input query and in each of the CQs of the rewriting.

Note that these CQs and ontologies have very few tree witnesses. More precisely, in 67% of the cases there are *no tree witnesses* at all, and in 29% we have only one. Even for the specially designed q_8, the structure of tree witnesses is simpler than in our example from Section 2.1 (e.g., they do not overlap). And although q_8 and q_9 do have tree witnesses, the resulting UCQs contain only one CQ since these tree witnesses are generated by other atoms of the queries. In fact, all tree-witness rewritings in our experiments contain at most two CQs: one of them is an optimised original CQ (in particular, by the domain/range optimisation (4) in s_2–s_5, r_2–r_5, q_1, q_3, q_5–q_8) and the other is obtained by replacing the atoms of the tree-witness with its generator. Thus, each of the CQs in the rewritings is not larger than the input query and has a very similar structure.

To illustrate, consider the following subquery of q_8:

$$\boldsymbol{q}_0(x_0) \leftarrow \mathit{Publication}(x_0),\ \mathit{publicationAuthor}(x_0, x_{11}),\ \mathit{Subj1Professor}(x_{11}),\\ \mathit{worksFor}(x_{11}, x_{12}),\ \mathit{Department}(x_{12}),$$

Table 2. The size of rewritings over $\text{LUBM}^{\exists}_{20}$ (DNF = Did Not Finish in 600s)

	r_1	r_2	r_3	r_4	r_5	q_1	q_2	q_3	q_4	q_5	q_6	q_7	q_8	q_9
UCQ (number of CQs)														
Requiem	2	1	23	2	10	DNF	2	DNF	14,880	690	DNF	DNF	DNF	DNF
Nyaya	2	1	23	2	10	DNF	2	DNF	DNF	690	DNF	DNF	DNF	DNF
IQAROS	2	1	23	2	10	DNF	**1**	15,120	14,400	690	23,552	DNF	DNF	DNF
Rapid	2	1	23	2	10	3,887	2	15,120	14,880	690	23,552	DNF	1	16
datalog (number of non-taxonomical rules)														
Rapid	1	1	1	1	1	2	3	1	2	1	1	1	27	1
Clipper	1	1	1	1	1	8	7	1	5	1	1	1	512	16
tw-rewriter	1	1	1	1	1	2	2	1	2	1	1	1	1	1

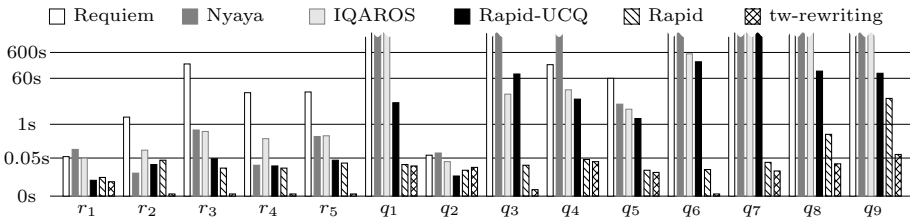


Fig. 1. Rewriting time for queries over $\text{LUBM}^{\exists}_{20}$

where x_{11}, x_{12} do not occur in the rest of q_8. This CQ has a tree witness comprising the last two atoms because of the $\text{LUBM}^{\exists}_{20}$ axiom $\mathit{Faculty} \sqsubseteq \exists \mathit{worksFor}$. However, *Subj1Professor* is a subclass of *Faculty*, and so any of its instances is always connected to *Department* by *worksFor* (either in the ABox or in the anonymous part). Thus, the last two atoms of $\boldsymbol{q}_0$ do not affect its answers and can be removed. The first atom is redundant by (4) with the domain axiom $\exists \mathit{publicationAuthor} \sqsubseteq \mathit{Publication}$, which results in the following rewriting: $\boldsymbol{q}'_0(x_0) \leftarrow \mathit{publicationAuthor}(x_0, x_{11}),\ \mathit{Subj1Professor}(x_{11})$. As $\boldsymbol{q}_0$ represents a natural and common pattern for expressing queries—select a *Publication* whose *publicationAuthor* is a *Subj1Professor*, etc.—any OBDA system should be able to detect such redundancies automatically.

For comparison, we computed the rewritings of the CQs over $\text{LUBM}^{\exists}_{20}$ using Requiem [20], Nyaya [9], IQAROS (v 0.2) [29], Rapid (v 0.3) [5] and Clipper (v 0.1) [7]. The first four return UCQ rewritings, the numbers of CQs in which are shown in Table 2. The last two return nonrecursive datalog rewritings over *arbitrary* ABoxes. These rewritings consist of a number of 'main' rules and a number of taxonomical rules for completing the ABoxes by subclasses/subproperties; to compare with *Ontop*, Table 2 shows only the number of the 'main' rules. Interestingly, Clipper and Rapid return single-rule rewritings in the cases without tree witnesses, but generate more rules than *Ontop* (e.g., q_8 and q_9) otherwise.

Figure 1 shows the time required for rewriting (it was impossible to separate rewriting from DLV execution in Clipper, but it terminated within 1.5s on every query). The UCQ-based systems do not finish in many cases and

require a substantial amount of memory (up to 1GB in some cases). In contrast, the datalog-based systems and *Ontop* produce rewritings very quickly. Observe that the rewritings returned by the four UCQ-based systems can be obtained from the tree-witness rewritings by replacing each class/property with its subclasses/subproperties (IQAROS's rewritings of q_2 and possibly q_4 are incorrect): for instance, q_7 gives 216,000 ($= 30^3 \times 2^3$) CQs, q_3 gives 15,120 ($=4 \times 5 \times 21 \times 36$) CQs and q_1 gives 3,887 ($= 23 + 2 \times 4 \times 21 \times 23$) CQs as *Student*, *Faculty* and *Professor* have 23, 36 and 30 subclasses, respectively, *worksFor* has 2 subproperties, etc. Such an operation (if needed) could be performed in fractions of seconds.

The experiments reported in this section imply that dealing efficiently with class/property hierarchies is the most critical component of any OBDA system. We discuss how *Ontop* copes with this task in the next section.

3.2 $\mathcal{T}$-mappings: Class and Property Hierarchies

We compare the query execution time in *Ontop*, Stardog 1.2 [21] and OWLIM [3]. Both Stardog and OWLIM use internal data structures to store RDF triples. Stardog is based on rewriting into UCQs (as we saw above, such systems can run out of memory during the rewriting stage, even before accessing the data). OWLIM is based on inference materialisation (forward chaining); but the implemented algorithm is known to be incomplete for OWL 2 QL [3].

It was impossible to compare *Ontop* with other systems: Rapid and IQAROS are just query rewriting algorithms; Clipper (v 0.1) supports only the DLV datalog engine that reads queries and triples at the same time (which would be a serious disadvantage for large datasets). The experiments were run on an HP Proliant with 24 Intel Xeon 6-core 3.47GHz CPUs, 106GB RAM and a 1TB@15000rpm HD under 64-bit Ubuntu 12.04 with Java 7, MySQL 5.6 and DB2 10.1.

Data as Triples: The Semantic Index. We first compare the performance of the three systems for the case where the data is stored in the form of triples. In this case, *Ontop* uses universal tables, and the SQO optimisations do not play any role. We took $\text{LUBM}^{\exists}_{20}$ with the data created by the modified LUBM data generator [19] for 50, 200 and 1000 universities (5% incompleteness) with 7m, 29m and 143m triples, respectively.

OWLIM requires a considerable amount of time for loading and materialising the inferences—14min, 1h 23min and 8h 4min, respectively—expanding the data by 93% and obtaining 13m, 52m and 252m triples. Neither Stardog nor *Ontop* need this expensive loading stage. The results of executing the queries from Section 3.1 are given in Table 3 (in order to reduce the influence of the result size, which are quite large in some cases, we executed queries that counted the number of distinct tuples rather than returned the tuples themselves). We note first that Stardog runs out of memory on 50% of the queries, with a likely cause being the query rewriting algorithm, which is an improved version of Requiem (cf. Table 2). On the remaining queries, Stardog is fast, which is probably due to its optimised triple store. Unlike Stardog, both OWLIM and *Ontop* return

Table 3. Query execution time (in seconds) and the result size over $\text{LUBM}^{\exists}_{20}$

		r_1	r_2	r_3	r_4	r_5	q_1	q_2	q_3	q_4	q_5	q_6	q_7	q_8	q_9
							50 universities								
Ontop	DB2	**0**	**0.03**	**0.50**	**0.01**	**0**	25.2	0.47	0.39	**0.04**	**1.37**	**0.07**	0.51	**0.13**	**0**
Ontop	MySQL	**0**	0.19	3.76	0.08	**0**	31.0	2.48	10.54	0.13	4.22	2.19	0.48	**0.13**	**0**
OWLIM		0.01	0.78	2.43	0.28	0.17	**12.9**	2.68	**0.21**	0.29	3.95	0.78	**0.23**	0.23	0.04
Stardog		0.01	0.79	1.16	0.34	0.10	DNF	**0.10**	DNF	DNF	DNF	DNF	DNF	DNF	0.04
result size		–	102k	12k	34k	–	1.2m	–	89	–	205k	–	–	–	–
							200 universities								
Ontop	DB2	**0**	**0.08**	7.33	**0.07**	**0**	522.9	1.75	3.48	**0.12**	**5.52**	**0.26**	**0.86**	**0.25**	**0**
Ontop	MySQL	**0**	1.21	14.6	0.32	**0**	260.4	9.30	34.02	0.49	16.11	8.45	1.66	0.54	**0**
OWLIM		0.01	3.10	9.28	0.94	0.79	**46.4**	10.52	**0.89**	15.15	16.91	3.32	0.92	0.92	0.05
Stardog		0.01	3.22	**2.92**	1.12	0.27	DNF	**0.33**	DNF	DNF	DNF	DNF	DNF	DNF	0.06
result size		–	410k	48k	137k	–	4.6m	–	399	–	825k	–	–	–	–
							1000 universities								
Ontop	DB2	**0**	**0.24**	**11.6**	**0.19**	**0**	2761	3.29	11.3	**0.58**	**12.7**	**1.15**	**5.38**	**1.23**	**0**
Ontop	MySQL	**0**	1.54	70.9	0.85	**0**	1232	90.9	185.3	2.37	132.7	2.86	7.75	2.48	**0**
OWLIM		0.01	18.9	63.6	6.40	3.38	**308**	65.7	**5.11**	94.3	105.7	2.76	5.84	5.79	0.10
Stardog		0.21	20.7	13.7	9.36	1.11	DNF	**3.07**	DNF	DNF	DNF	DNF	DNF	DNF	0.17
result size		–	2m	239k	685k	–	23m	–	2k	–	4.1m	–	–	–	–

answers to all queries, and their performance is comparable. In fact, in 83% of the cases *Ontop* with DB2 outperforms OWLIM.

It is to be emphasised that *Ontop* can work with a variety of database engines and that, as these experiments demonstrate, *Ontop* with MySQL in many case is worse in executing queries than with DB2 (but is still competitive with OWLIM). Two techniques turned out to be crucial to improve the performance of the engines. First, in the universal relations $CA[id, \textit{class-id}]$ and $RA[id_1, id_2, \textit{property-id}]$, we store integer URI identifiers rather than URIs themselves, with a special relation $URI[id, uri]$ serving as a dictionary to de-reference the URI identifiers. Second, a significant improvement of performance was achieved by creating indexes on sequences of attributes of the universal relations: for example, CA has indexes on $(id, \textit{class-id})$, (id) and $(\textit{class-id})$. The full impact of such indexes on storing data in the form of RDF triples is yet to be investigated.

Finally, we observe that some queries do not need evaluation because *Ontop* simplifies them to empty queries: in fact, r_1, r_5 and q_9 contain atoms that have no instances in the generated data, and only 6 out of the 14 CQs return any answers (which probably reflects the artificial nature of the benchmark).

These experiments confirm once again that rewritings into UCQs over arbitrary ABoxes can be prohibitively large even for high-performance triple stores such as Stardog. The materialisation approach should 'by definition' cope with large taxonomies. We have demonstrated that the semantic index used in *Ontop* is able to deal with this problem as efficiently as (and often better than) inference materialisation, without the considerable overhead expense of the latter.

Table 4. Query and rewriting metrics, result sizes and execution times (in seconds)

	q_1	q_2	q_3	q_4	q_5	q_6	q_7	q_8	q_9	q_{10}
atoms in query	4	15	6+f	4+f	9	6	8	6	9+f	3+f
UCQ rewriting	2	48	2	1	24	2	4	16	4	1
tables in SQL	3	13	3	2	6	4	5	4	8	2
result size	6	14,688	15,010	2,224	1,921	84	4	59,211	48	26
Stardog result size	0	0	4,047	0	9	0	0	27,804	0	26
Ontop-DB2	**0.003**	0.626	**0.495**	**0.355**	7.525	**0.005**	**0.001**	**0.699**	0.167	**0.358**
Ontop-MySQL	0.005	6.138	0.679	0.571	9.190	0.009	0.023	3.563	0.460	0.457
OWLIM	0.005	**0.562**	5.413	2.833	**0.681**	0.009	0.007	4.307	**0.046**	0.836
Stardog	0.030	1.136	1.329	2.227	1.389	0.029	0.038	1.277	0.409	0.584

***Ontop* of Databases.** We now evaluate the performance of the $\mathcal{T}$-mapping approach to answering queries over OWL 2 QL ontologies with mappings to real-world databases. We use the Movie Ontology (MO, `www.movieontology.org`) and the data from the SQL version of the Internet Movie Database (IMDb, `www.imdb.com/interfaces`). Both the database and ontology were developed independently by third parties for purposes different from benchmarking; the mapping was created by the *Ontop* development team. MO has 137 class and property names and 157 inclusion axioms; the mapping contains 271 rules and the virtual ABox has 42m assertions. We tested 10 natural queries to IMDb: e.g., q_3 retrieves the companies from East Asia and the movies they produced between 2006 and 2010.

The metrics of the queries and their rewritings, the numbers of returned tuples, and the execution times by *Ontop* with DB2 and MySQL, OWLIM and Stardog over the materialised ABox are shown in Table 4. The line 'atoms in query' gives the number of atoms in the input query (+f denotes a FILTER expression). Each query coincides with its tree-witness rewriting (there are no tree witnesses, and none of the atoms is redundant). The line 'UCQ rewriting' shows the number of CQs in the rewritings over *arbitrary* ABoxes, which reflects the size of class and property hierarchies. The resulting SQL query contains a single SELECT-PROJECT-JOIN component with the number of tables given by 'tables in SQL'—this corresponds to the number of JOINs in the SQL query. Because of the SQO, the SQL queries have fewer tables and JOINs than the original one (or the rewriting). For example, q_3 with 6 atoms produces a single SPJ query with 3 tables (and one disjunction over 7 country codes rather than 7 subqueries):

```
SELECT DISTINCT Q3.name, Q1.title, Q1.production_year
FROM title Q1, movie_companies Q2, company_name Q3
WHERE (Q1.id = Q2.movie_id) AND (Q2.company_id = Q3.id) AND
  (('[tw]' = Q3.country_code) OR ... OR ('[kr]' = Q3.country_code)) AND
  (Q1.production_year <= 2010) AND (Q1.production_year >= 2006)
```

Note that Stardog, on the same set of triples as OWLIM, returns *fewer* tuples in *all cases* but q_{10}, which may explain the better execution times (one of the Stardog optimisations [21] removes empty CQs from the rewriting and may be responsible for the missing tuples).

In 70% cases, *Ontop* with DB2 outperforms OWLIM (and is efficient even with MySQL). Moreover, OWLIM takes 45min to load the data into the triple store (and will have to do this again every time the data is changed). This demonstrates that on-the-fly inference over real-world databases by means of the tree-witness rewriting and $\mathcal{T}$-mappings is efficient enough to successfully compete with materialisation-based techniques. Moreover, the usual problems associated with query-rewriting-based approaches disappear in *Ontop*: $\mathcal{T}$-mappings efficiently deal with hierarchical reasoning avoiding the exponential blowup, and the SQO improves the performance of the produced SQL queries by taking account of the structure and integrity constraints of the database.

4 Conclusions

To conclude, we believe this paper shows that—despite the negative theoretical results on the worst-case OWL 2 QL query rewriting and sometimes disappointing experiences of the first OBDA systems—high-performance OBDA is achievable in practice when applied to real-world ontologies, queries and data stored in relational databases. In such cases, query rewriting together with SQO and SQL optimisations is fast, efficient and produces SQL queries of high quality.

Acknowledgements. We thank G. Orsi for providing Nyaya, G. Xiao for providing queries and the *Ontop* development team (J. Hardi, T. Bagosi, M. Slusnys) for the help with the experiments. This work was supported by the EU FP7 project Optique (grant 318338) and UK EPSRC project ExODA (EP/H05099X).

References

1. Abiteboul, S., Hull, R., Vianu, V.: Foundations of Databases. Addison-Wesley (1995)
2. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F. (eds.): The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press (2003)
3. Bishop, B., Bojanov, S.: Implementing OWL 2 RL and OWL 2 QL rule-sets for OWLIM. In: Proc. of OWLED 2011, vol. 796. CEUR-WS (2011)
4. Chakravarthy, U.S., Fishman, D.H., Minker, J.: Semantic query optimization in expert systems and database systems. Benjamin-Cummings Publ. Co., Inc. (1986)
5. Chortaras, A., Trivela, D., Stamou, G.: Optimized query rewriting for OWL 2 QL. In: Bjørner, N., Sofronie-Stokkermans, V. (eds.) CADE 2011. LNCS (LNAI), vol. 6803, pp. 192–206. Springer, Heidelberg (2011)
6. Dolby, J., Fokoue, A., Kalyanpur, A., Ma, L., Schonberg, E., Srinivas, K., Sun, X.: Scalable grounded conjunctive query evaluation over large and expressive knowledge bases. In: Sheth, A.P., Staab, S., Dean, M., Paolucci, M., Maynard, D., Finin, T., Thirunarayan, K. (eds.) ISWC 2008. LNCS, vol. 5318, pp. 403–418. Springer, Heidelberg (2008)
7. Eiter, T., Ortiz, M., Šimkus, M., Tran, T.K., Xiao, G.: Query rewriting for Horn-SHIQ plus rules. In: Proc. of AAAI 2012. AAAI Press (2012)
8. Elmasri, R., Navathe, S.: Fundamentals of Database Systems, 6th edn. Addison-Wesley (2010)

9. Gottlob, G., Orsi, G., Pieris, A.: Ontological queries: Rewriting and optimization. In: Proc. of ICDE 2011, pp. 2–13. IEEE Computer Society (2011)
10. Gottlob, G., Schwentick, T.: Rewriting ontological queries into small nonrecursive datalog programs. In: Proc. of KR 2012. AAAI Press (2012)
11. Heymans, S., Ma, L., Anicic, D., Ma, Z., Steinmetz, N., Pan, Y., Mei, J., Fokoue, A., Kalyanpur, A., Kershenbaum, A., Schonberg, E., Srinivas, K., Feier, C., Hench, G., Wetzstein, B., Keller, U.: Ontology reasoning with large data repositories. In: Ontology Management, Semantic Web, Semantic Web Services, and Business Applications, pp. 89–128. Springer (2008)
12. Kementsietsidis, A., Bornea, M., Dolby, J., Srinivas, K., Dantressangle, P., Udrea, O., Bhattacharjee, B.: Building an efficient RDF store over a relational database. In: Proc. of SIGMOD 2013. ACM (2013)
13. Kikot, S., Kontchakov, R., Podolskii, V., Zakharyaschev, M.: Exponential lower bounds and separation for query rewriting. In: Czumaj, A., Mehlhorn, K., Pitts, A., Wattenhofer, R. (eds.) ICALP 2012, Part II. LNCS, vol. 7392, pp. 263–274. Springer, Heidelberg (2012)
14. Kikot, S., Kontchakov, R., Podolskii, V., Zakharyaschev, M.: Query rewriting over shallow ontologies. In: Proc. of DL 2013, vol. 1014. CEUR-WS (2013)
15. Kikot, S., Kontchakov, R., Zakharyaschev, M.: Conjunctive query answering with OWL 2 QL. In: Proc. of KR 2012. AAAI Press (2012)
16. König, M., Leclère, M., Mugnier, M.-L., Thomazo, M.: A sound and complete backward chaining algorithm for existential rules. In: Krötzsch, M., Straccia, U. (eds.) RR 2012. LNCS, vol. 7497, pp. 122–138. Springer, Heidelberg (2012)
17. Kontchakov, R., Lutz, C., Toman, D., Wolter, F., Zakharyaschev, M.: The combined approach to query answering in DL-Lite. In: Proc. of KR 2010. AAAI (2010)
18. Lloyd, J., Shepherdson, J.: Partial Evaluation in Logic Programming. The Journal of Logic Programming 11(3-4), 217–242 (1991)
19. Lutz, C., Seylan, İ., Toman, D., Wolter, F.: The combined approach to OBDA: Taming role hierarchies using filters. In: Proc. of SSWS+HPCSW 2012 (2012)
20. Pérez-Urbina, H., Motik, B., Horrocks, I.: A comparison of query rewriting techniques for DL-lite. In: Proc. of DL 2009, vol. 477. CEUR-WS (2009)
21. Pérez-Urbina, H., Rodríguez-Díaz, E., Grove, M., Konstantinidis, G., Sirin, E.: Evaluation of query rewriting approaches for OWL 2. In: Proc. of SSWS+HPCSW 2012, vol. 943. CEUR-WS (2012)
22. Poggi, A., Lembo, D., Calvanese, D., De Giacomo, G., Lenzerini, M., Rosati, R.: Linking data to ontologies. In: Spaccapietra, S. (ed.) Journal on Data Semantics X. LNCS, vol. 4900, pp. 133–173. Springer, Heidelberg (2008)
23. Rodríguez-Muro, M.: Tools and Techniques for Ontology Based Data Access in Lightweight Description Logics. Ph.D. thesis, Free Univ. of Bozen-Bolzano (2010)
24. Rodríguez-Muro, M., Calvanese, D.: Dependencies: Making ontology based data access work. In: Proc. of AMW 2011, vol. 749. CEUR-WS (2011)
25. Rodríguez-Muro, M., Kontchakov, R., Zakharyaschev, M.: Query rewriting and optimisation with database dependencies in Ontop. In: Proc. of DL 2013 (2013)
26. Rosati, R.: Prexto: Query rewriting under extensional constraints in DL-Lite. In: Simperl, E., Cimiano, P., Polleres, A., Corcho, O., Presutti, V. (eds.) ESWC 2012. LNCS, vol. 7295, pp. 360–374. Springer, Heidelberg (2012)
27. Rosati, R., Almatelli, A.: Improving query answering over DL-Lite ontologies. In: Proc. of KR 2010. AAAI Press (2010)
28. Sequeda, J., Miranker, D.: Ultrawrap: SPARQL execution on relational data. Tech. Rep. TR-12-10, Dept. of Computer Science, University of Texas at Austin (2012)
29. Venetis, T., Stoilos, G., Stamou, G.: Query extensions and incremental query rewriting for OWL 2 QL ontologies. Journal on Data Semantics (2013)
30. Xiao, G.: Personal communication (2013)

DAW: Duplicate-AWare Federated Query Processing over the Web of Data

Muhammad Saleem[1,⋆], Axel-Cyrille Ngonga Ngomo[1], Josiane Xavier Parreira[2], Helena F. Deus[2], and Manfred Hauswirth[2]

[1] Universität Leipzig, IFI/AKSW, PO 100920, D-04009 Leipzig
lastname@informatik.uni-leipzig.de
[2] Digital Enterprise Research Institute, National University of Ireland, Galway
firstname.lastname@deri.org

Abstract. Over the last years the Web of Data has developed into a large compendium of interlinked data sets from multiple domains. Due to the decentralised architecture of this compendium, several of these datasets contain duplicated data. Yet, so far, only little attention has been paid to the effect of duplicated data on federated querying. This work presents DAW, a novel duplicate-aware approach to federated querying over the Web of Data. DAW is based on a combination of min-wise independent permutations and compact data summaries. It can be directly combined with existing federated query engines in order to achieve the same query recall values while querying fewer data sources. We extend three well-known federated query processing engines – DARQ, SPLENDID, and FedX – with DAW and compare our extensions with the original approaches. The comparison shows that DAW can greatly reduce the number of queries sent to the endpoints, while keeping high query recall values. Therefore, it can significantly improve the performance of federated query processing engines. Moreover, DAW provides a source selection mechanism that maximises the query recall, when the query processing is limited to a subset of the sources.

Keywords: federated query processing, SPARQL, min-wise independent permutations, Web of Data.

1 Introduction

The emergence of the Web of Data has resulted in a large compendium of interlinked datasets from multiple domains available on the Web. The central principles underlying the architecture of these datasets include the decentralized provision of data, the reuse of URIs and vocabularies, as well as the linking of knowledge bases [2]. As a result, certain queries can only be answered by retrieving information from several data sources. This type of queries, called *federated queries*, are becoming increasingly popular within the Web of Data [1,3,8,9,12,14,21,22]. Recently, the W3C released the SPARQL 1.1 specification which directly addresses federated queries [1]. Due to the independence of

⋆ This work was carried out while the author was a research assistant in DERI.
[1] http://www.w3.org/TR/sparql11-federated-query/

H. Alani et al. (Eds.): ISWC 2013, Part I, LNCS 8218, pp. 574–590, 2013.

the data sources, certain pieces of information (i.e., RDF triples) can be found in multiple data sources. For example, all triples from the *DrugBank*[2] and *Neurocommons*[3] datasets can also be found in the *DERI health Care and Life Sciences Knowledge Base*[4]. We call triples that can be found in several knowledge bases *duplicates*.

While the importance of federated queries over the Web of Data has been stressed in previous work, the impact of duplicates has not yet received much attention. Recently, the work in [11] presented a benefit-based source selection strategy, where the benefit of a source is inversely proportional to the overlap between the source's data and the results already retrieved. The overlap is computed by comparing data summaries represented as Bloom filters [5]. The approach follows an "index-free" paradigm, and all the information about the sources is obtained at query time, for each triple pattern in the query.

In this paper we present DAW, a duplicate-aware approach for federated query processing over the Web of Data. Similar to [11] our approach uses sketches to estimate the overlap among sources. However, we adopt an "index-assisted" approach, where compact summaries of the sources are pre-computed and stored. DAW uses a combination of min-wise independent permutations (MIPs) [6] and triple selectivity information to estimate the overlap between the results of different sources. This information is used to rank the data sources, based on how many new query results are expected to be found. Sources that fall below a predefined threshold are discarded and not queried.

We extend three well-known federated query engines – DARQ [21], SPLENDID [8], and FedX [22] – with DAW, and compare these extensions with the original frameworks. The comparison shows that DAW requires fewer sources for each of the query's triple pattern, therefore improving query execution times. The impact on the query recall due to the overlap estimation was minimal, and in most cases the recall was not affected. Moreover, DAW provides a source selection mechanism that maximises the query recall when the query processing is limited to a subset of the sources.

The rest of this paper is zed as follows: Section 2 describes the state-of-the-art in federated query processing and different statistical synopsis approaches that can be used for approximating duplicate-free result sets. Section 3 describes our novel duplicate-aware federated query processing approach. An evaluation of DAW against existing federated query approaches is given in Section 4. Finally, Section 5 concludes our paper and presents directions for future work.

2 Related Work

In recent years, many approaches have been proposed for federated query processing for the Web of Data. Quilitz and Leser [21] propose an index-assisted federated query engine named DARQ for remote RDF data sources.

[2] `http://datahub.io/dataset/fu-berlin-drugbank`
[3] `http://neurocommons.org/page/RDF_distribution`
[4] `http://hcls.deri.org:8080/openrdf-sesame/repositories/hclskb`

DARQ combines service descriptions, query rewriting mechanisms and a cost-based optimisation approach to reduce the query processing time and the bandwidth usage. Langegger et al. [13] describe a solution similar to DARQ that relies on a mediator to keep its service descriptions up-to-date. SPLENDID [8] uses VOID[5] descriptions for data source selection along with SPARQL *ASK* queries. All of the approaches described above can be considered to be index-assisted, since they all rely in some sort of local index to guide the source selection process. Index-free approaches include FedX [22] and the Avalanche system [3]. In FedX, the source selection is performed by using *ASK* queries, while Avalanche gathers endpoints dataset statistics and bandwidth availability on the fly before the query federation. Ludwig and Tran [12] propose a hybrid query engine that assumes some incomplete knowledge about the sources to select and discover new sources at run time. A symmetric hash join is used to incrementally produce answers. Acosta et al. [1] present ANAPSID, a query engine that adapts the query execution schedulers to the SPARQL endpoints' data availability and run-time conditions.

Overlap estimation among data sources have been used in a number of approaches in the area of distributed and P2P information retrieval [4,10,15,18,23,24]. COSCO [10] gathers statistics about coverage and overlap from past queries and uses them to determine in which order the overlapping collections should be accessed to retrieve the most new results in the least number of collections. Bender et al. [4] describes a novelty estimator that uses Bloom filters [5] to estimate the overlap between P2P data sources. Bloom filters are also used in the BBQ strategy for benefit-based query routing over federated sources [11].

Statistical synopsis such as Min-Wise Independent Permutations (MIPs) [6], Bloom filters [5], Hash sketches [19], XSKETCH [20], fractional XSKETCH [7], and compressed Bloom filters [16] have been extensively used in the literature to provide a compacted representation of data sets. MIPs have been shown to be the provide a good tradeoff between estimation error and space requirements [15,6]. In addition, MIPs of different lengths can be compared, which can be beneficial for datasets of different sizes.

3 Duplicate-Aware Federated Query Processing

In this section we present our DAW approach. DAW can be used in combination with existing federated query processing systems to enable a duplicate-aware query execution.

Given a SPARQL query q, the first step is to perform a *triple pattern-wise source selection*, i.e., to identify the set of data sources that contain relevant results for each of the triple patterns of the query. This is done by the underlying federated system. For a given triple pattern, the relevant sources are also called *capable* sources. The idea of DAW federated query processing is, for each triple pattern and its set of capable sources, to (i) *rank* the sources based on how

[5] `http://www.w3.org/TR/void/`

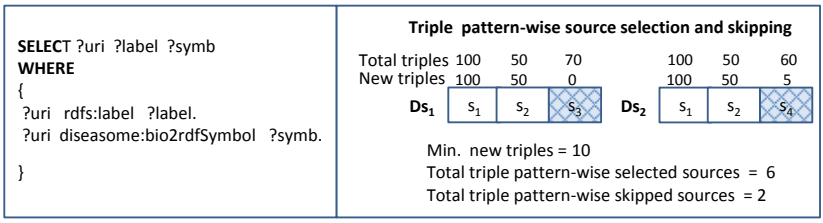


Fig. 1. Triple pattern-wise source selection and skipping example

much they can contribute with *new* query results, and (ii) *skip* sources which are ranked below a predefined threshold. We call these two steps *triple pattern-wise source ranking* and *triple-pattern wise source skipping*. After that, the query and the list of not skipped sources are forwarded to the underlying federated query engine. The engine generates the subqueries that are sent to the relevant SPARQL endpoints. The results of each subquery execution are then joined to generate the result set of q.

To better illustrate this, consider the example given in Figure 1, which shows a query with two triple patterns (tp_1 and tp_2), and the lists of capable sources for both patterns. For each source we show the total number of triples containing the same predicate of the triple pattern and the estimated number of new triples, i.e. triples that do not overlap with the previous sources in the list. The triple pattern-wise source ranking step orders the sources based on their contribution. As we see in the example, for the triple pattern tp_1, source S_1 is ranked first, since it is estimated to produce 100 results. S_1 is followed by S_2, which can contribute with 40 new results, considering the overlap between the two sets. S_3 is ranked last, despite having more triples than S_2. This is because our duplicated-aware estimation could not find any triple in S_3 which is not in either S_1 or S_2. In the triple-pattern wise source skipping step, S_3 will be discarded, and tp_1 will not be sent to S_3 during query execution. We can also set a threshold on the minimum number of results. For instance, by setting the threshold to 10 results, source S_4 will be skipped, since it can only contribute with 5 new results for tp_2. By applying our duplicate-aware approach – which would select S_1 and S_2 both for *tp1* and *tp2* and would skip S_3 and S_4 – we would only send subqueries to two endpoints instead of four.

Both steps are performed prior to the query execution, by using only information contained in the DAW index. The main innovation behind DAW is to avoid querying sources which would lead to duplicated results. We achieve this by extending the idea of min-wise independent permutations (MIPs) [6], which are explained in the next section.

3.1 Min-Wise Independent Permutations (MIPs)

The main rationale behind MIPs is to enable the representation of large sets as vectors of smaller magnitude and to allow the estimation of a number of set

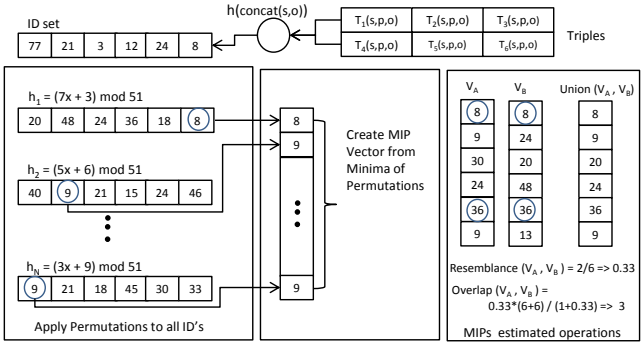


Fig. 2. Min-Wise Independent Permutations

operations, such as overlap and union, without having to compare the original sets directly. The basic assumption behind MIPs is that each element of an ordered set S has the same probability of becoming the minimum element under a random permutation. MIPs assumes an ordered set S as input and computes N random permutations of the elements. Each permutation uses a linear hash function of the form $h_i(x) := a_i * x + b_i \ mod \ U$ where U is a big prime number, x is a set element, and a_i, b_i are fixed random numbers. By ordering the set of resulting hash values, we obtain a random permutation of the elements of S. For each of the N permutations, the MIPs technique determines the minimum hash value and stores it in an N-dimensional vector, thus capturing the minimum set element under each of these random permutations. The technique is illustrated in Figure 2.

Let $V_A = [a_1, a_2, \ldots, a_N]$ and $V_B = [b_1, b_2, \ldots, b_N]$ be the two MIPs vectors representing two ordered ID's sets S_A, S_B, respectively. An unbiased estimate of the pair-wise resemblance between the two sets, i.e. the fraction of elements that both sets share with each other, is obtained by counting the number of positions in which the two MIPs vectors have the same number and dividing this by the number of permutations N as shown in Equation 1. It can be shown that the expected error in the estimation $O(1/\sqrt{N})$ [6]. Given the resemblance and the sizes of the two set, their overlap can be estimated as shown in Equation 2. A MIPs vector representing the union of the two sets, S_A and S_B, can be created directly from the individuals MIPs vectors, V_A and V_B, by comparing the pair-wise entries, and storing the minimum of the two values in the resulting union vector (see Figure 2). A nice property of MIPs is that unions can be computed even if the two MIPs vectors have different sizes, as long as they use the same sequence of hash functions for creating their permutations. In general, if two MIPs have different sizes, we can always use the smaller number of permutations as a common denominator. This incurs in a loss of accuracy in the result MIPs, but still yields to a more flexible setting, where the different collections do not have to agree on a predefined MIPs size [15].

$$Resemblance(S_A, S_B) = \frac{|S_A \cap S_B|}{|S_A \cup S_B|} \approx \frac{|V_A \cap V_B|}{N} \tag{1}$$

$$Overlap(S_A, S_B) \approx \frac{Resemblance(V_A, V_B) \times (|S_A| + |S_B|)}{(Resemblance(V_A, V_B) + 1)} \quad (2)$$

In the DAW index, MIPs are used as follow: For a distinct predicate p belonging to a data source S, we define $T(p, S)$ as the set of all triples in S with predicate p. A MIPs vector is then created for every $T(p, S)$. First an *ID set* is generated by mapping each triple in $T(p, Sr)$ to an integer value. A triple is given in the form of subject, predicate and object tuples, i.e. $< s, p, o >$. Since all triples in $T(p, S)$ share the same predicate by definition, the mapping is done by concatenating the subject (s) and object (o) of the triple, and applying a hash function to it (Figure 2). Then, the MIPs vector is created by computing the N random permutations of each element in the *ID set* and storing their minimum value. Finally, the MIPs vector is stored and mapped to each capability of the service description, as explained in the next section.

3.2 DAW Index

In order to detect duplicate-free subqueries, DAW relies on an index which contains the following information for every distinct predicate p in a source S:

1. The total number of triples $n_S(p)$ with the predicate p in S.
2. The MIPs vector $MIPs_S(p)$ for the predicate p in S, as described in the previous section.
3. The *average subject selectivity* of p in S, $avgSbjSel_S(p)$.
4. The *average object selectivity* of p in S, $avgObjSel_S(p)$.

The average subject and object selectivities are defined as the inverse of the number of distinct subjects and objects which appears with predicate p, respectively. For example, given the following set of triples:

$$S = \{< s_1, p, o_1 >, < s_1, p, o_2 >, < s_2, p, o_1 >, < s_3, p, o_2 >\} \quad (3)$$

the $avgSbjSel_S(p)$ is equal to $\frac{1}{3}$ and the $avgObjSel_S(p)$ is $\frac{1}{2}$. These two values are used in combination with the MIPs vector to address the expressivity of SPARQL queries as explained below.

Suppose that in a given triple pattern, neither the subject nor the predicate are bound. That means the pattern is of the form $<?s, p, ?o >$, where the question mark denotes a variable. In this case, the MIPs vectors in the DAW index can be used directly to estimate the overlap among the data sources that can provide results for the pattern. This is because the MIPs vectors are created by grouping triples according to their predicate. However, if any of the subject or object is bound (for example, $< s_1, p, ?o >$), the selectivity of the pattern becomes much higher and the MIPs vectors alone are unable to address this. As a result, overlap will be overestimated. To address this issue the modify Equation 2 to account for the subject and object selectivities as follows:

$$Overlap_{tp}(S_A, S_B) \approx \frac{Resemblance(V_A, V_B) \times (|S'_A| + |S'_B|)}{(Resemblance(V_A, V_B) + 1)} \quad (4)$$

Listing 1.1. DAW index example

```
[] a sd:Service ;
     sd:endpointUrl   <http://localhost:8890/sparql> ;
     sd:capability [
      sd:predicate  diseasome:name ;
      sd:totalTriples     147  ;
      sd:avgSbjSel  ''0.0068'' ;
      sd:avgObjSel  ''0.0069'' ;
      sd:MIPs  ''-6908232 -7090543 -6892373 -7064247 ...''; ] ;
     sd:capability [
      sd:predicate  diseasome:chromosomalLocation ;
      sd:totalTtriples     160  ;
      sd:avgSbjSel  ''0.0062'' ;
      sd:avgObjSel  ''0.0072'' ;
      sd:MIPs  ''-7056448 -7056410 -6845713 -6966021 ...''; ] ;
```

where the original size of a set S_i is replaced by a value $|S_i'|$ which is given by the following equation:

$$|S_i'| = \begin{cases} |S_i| & \text{if neither subject nor object are bound,} \\ |S_i| \times avgSbjSel_S(p) & \text{if subject is bound,} \\ |S_i| \times avgObjSel_S(p) & \text{if object is bound.} \end{cases}$$

We call the set $C_S(p) = \{p, n_S(p), avgSbjsel_S(p), avgObjSel_S(p), MIPs_S(p)\}$ a *capability* of the data source. The total number of capabilities of a data source is equal to the number of distinct predicates in it.

It is crucial to keep the index size small to minimise the pre-processing time. On the other hand, this index must also contain sufficient information to enable an accurate source selection and duplicate-free subquery generation. Some federated query approaches such as DARQ and SPLENDID already provide the total number of triples, as well as the average selectivity values. Therefore, the storage overhead create by the DAW index depends mostly on the size of the MIPs vectors which can be adjusted to any length. In general, MIPs can provide a good estimation of the overlap between sets with a few integer in length. An example of a DAW index is given in Listing 1.1.

3.3 DAW Federated Query Processing

As explained earlier, given a SPARQL query, DAW performs the triple pattern-wise source ranking and skipping steps in order to rank the sources based on how much they can contribute with *new* query results, and skip sources which are below a given threshold. In this section we describe these two steps in detail.

Triple Pattern-Wise Source Ranking: Given the heterogeneity and independence of data sources, it is expected that each source contributed differently in answering a given triple pattern, and the same result might be returned by multiple sources. Our goal is to provide a rank of the sources, according to the estimated number of new results it can contribute. By new results we mean with respect to the results already retrieved from sources ranked higher.

The source ranking step works as follows: First, as no source has been ranked yet, the algorithm chooses the largest source, as it will likely to contribute with more results. To select the next source we use the DAW index to compute the estimated overlap between the already selected source and every remaining source. The remaining source with the least amount of overlap is then chosen and ranked second. Before selecting the next source in the rank, we first need to estimate the union of the already selected sources. This is needed since we want to find out how much a source can contribute with results are not in the sources selected so far. The union can be easily estimated by applying a vector operator on the original MIPs, as explained in Section 3.1. The new union MIPs can be further combined with other MIPs to get the estimation of the union among several sets. The source ranking step continues until no more sources are left to be ranked.

Triple Pattern-Wise Source Skipping: Given the rank of capable sources, the next step is to prune the rank, but skipping sources which cannot contribute with a minimum number of new results. This is done by setting a threshold, and pruning every source which falls below it. Since the total number of results depends on the triple pattern, the threshold is chosen in terms of the minimum percentage of new results a source can contribute. For instance, if the threshold is set to zero, DAW will aim at retrieving as much results as possible, while still skipping sources which cannot contribute with new results. Alternatively, the threshold can be set to higher values, in cases where the tradeoff between recall and number of sources queries is more important.

The pseudo code of the triple pattern-wise source ranking and skipping is given in Algorithm 1. It takes a triple pattern $tp_i(s, p, o)$, its list of capable sources $\mathbb{S}_i$, and the predefined threshold value as input and returned a ranked list of a subset of the capable source set R_i, $R_i \subseteq \mathbb{S}_i$ as output. The ranked list and the MPIs with the union of the selected sources are initialised with the largest source. Lines 8-14 adjust the size of the dataset to reflect the subject or object selectivities, depending on the query. Lines 15-16 estimate the overlap and number of new triples. The source with the highest amount of new triples is then selected (Lines 17-19). The triple pattern-wise source skipping is done in Line 23 and sources ranked higher than the threshold are added to the final ranked list (Line 24). The union MIPs is then updated (Line 26) and the algorithm continues until no more sources are left.

Before we present our experimental analysis of DAW it is important to note the difference between the number of triple pattern-wise sources and the number of sources (e.g. SPARQL endpoints). The total number of triple pattern-wise selected sources for a query is calculate as follow: Let $NS_i \in \{1 \dots M\}$ be the number of sources capable of answering a triple pattern tp_i where M is the number of available (physical) sources. Then, for a query q with n triple patterns, $\{tp_1, tp_2, \dots tp_n\}$, the total number of triple pattern-wise sources is the sum of the sources for individual triple patterns, i.e. $\sum_{j=1}^{n} NS_j$. In the example from Figure 1, the number of sources is 4 (s_1, s_2, s_3, s_4) but the number of triple pattern-wise sources is equal to 6.

Algorithm 1. Triple pattern source-wise ranking and skipping

Require: tp_i(s,p,o) ∈ T; $\mathbb{S}_i$; thresholdVal //triple pattern tp_i, capable data sources of tp_i; Threshold Value

```
1: rank1Source = getMaxSizeSource(S_i, tp_i) ; rnkNo = 1
2: unionMIPs = getMIPs(rank1Source, tp_i) //get MIP vector for a tp of a source
3: R_i[rnkNo] = selectedSource
4: S_i = S_i - {selectedSource}
5: rnkNo = rnkNo+1
6: while S_i ≠ ∅ do
7:    selectedSource = null; maxNewTriples =0
8:    for each S_i ∈ S_i do
9:       MIPs = getMIPs(S_i, tp_i)
10:      if s is bound in tp_i then
11:         MIPsSetSize =MIPsSetSize*getAvgSbjSel(S_i,tp_i)
12:      else if o is bound in tp_i then
13:         MIPsSetSize =MIPsSetSize*getAvgObjSel(S_i,tp_i)
14:      end if
15:      overlapSize = Overlap(unionMIPs,MIPs)
16:      newTriples = unionMIPsSetSize - overlapSize
17:      if newTriples > maxNewTriples then
18:         selectedSource = S_i
19:         maxNewTriples = newTriples
20:      end if
21:   end for
22:   curThresholdVal = unionMIPsSetSize / maxNewTriples
23:   if curThresholdVal ⩾ thresholdVal then
24:      R_i[rnkNo] = selectedSource
25:      selectedMIPs = getMIPs(selectedSource, tp_i)
26:      unionMIPs = Union(unionMIPs,selectedMIPs)
27:      rnkNo = rnkNo+1
28:   end if
29:   S_i = S_i - {selectedSource}
30: end while
31: return R_i //ranked list of capable sources for tp_i
```

4 Experimental Evaluation

In this section we present an experimental evaluation of the DAW approach. We first describe the experimental setup, followed by the evaluation results. All data used in this evaluation can be found at the project web page.[6]

4.1 Experimental Setup

Datasets: For our experiments, we used four different datasets. The Diseasome dataset contains diseases and disease genes linked by disease-gene associations.

[6] `https://sites.google.com/site/DAWfederation/`

Table 1. Overview of the datasets used in the experiments

Dataset	Number Triples	Dataset Size (MB)	Index Size (MB)	Index. Gen. Time (sec)	Discrepancy	No. Duplicated Slices	Duplicate Slice ID
Diseasome	91,122	18.6	0.17	4	1,500	1	10
Publication	234,405	39.0	0.24	6	2,500	1	10
Geo	1,900,006	274.1	1.63	133	50,000	2	5,8
Movie	3,579,616	448.9	1.66	201	100,000	1	2

Table 2. SPARQL endpoints specification

EP	CPU(GHz)	RAM	Hard Disk
1	2.2, i3	4GB	300 GB
2	2.9, i7	16 GB	256 GB SSD
3	2.6, i5	4 GB	150 GB
4	2.53, i5	4 GB	300 GB
5	2.3, i5	4 GB	500 GB
6	2.53, i5	4 GB	300 GB
7	2.9, i7	8 GB	450 GB
8	2.6, i5	8 GB	400 GB
9	2.6, i5	8 GB	400 GB
10	2.9, i7	16 GB	500 GB

Table 3. Distribution of query types across datasets

Dataset	STP	S-1	S-2	P-1	P-2	P-3	Total
Diseasome	5	5	5	4	5	2	26
Geo	5	5	5	-	-	-	15
Movie	5	-	-	-	-	-	5
Publication	5	5	5	7	7	4	33
Total	20	15	15	11	12	6	79

The Publication dataset is the Semantic Web Dog Food dataset and contains information on publications, venues and authors of publications. The Geo dataset resulted from retrieving the portion of triples from DBpedia that maps resources to their geo-coordinates. Finally, the Movie dataset is the RDF version of IMDB and contains amongst others a large number of actors, movies and directors. To simulate a federated scenario with fragmented datasets distributed across several sources, we partitioned each dataset in 10 slices and distributed the slices across 10 data sources (one slice per data source). Each data source is a Virtuoso-2012-08-02 SPARQL endpoint with the specifications given in Table 2.

To distribute the data across our 10 endpoints we defined a *discrepancy factor*, which controls the maximal size difference between the different slices.

$$discrepancy = \max_{1 \leq i \leq M} |L_i| - \min_{1 \leq j \leq M} |L_j|, \tag{5}$$

where L_i stands for the ith slice. The data is first partitioned randomly among the slices in a way that $\sum_i |L_i| = D$ and $\forall i \forall j \, i \neq j \rightarrow ||L_i| - |L_j|| \leq discrepancy$.

None of the existing benchmarks for federated query processing addresses the data duplication issue. Therefore, in order to add duplicates among slices, we randomly selected a number of slices and duplicated their contents across all remaining slices. For the DAW index, we use MIPs vectors of different sizes to better reflect the number of triples per predicate in each source. The sizes were chosen in a way that the overall index size is kept small. Table 1 presents an overview of the datasets, including the total number of triples and total size, the

size of the DAW index, the index generation time, the discrepancy value among the 10 slices, the number of slices that were duplicated and their corresponding ID.

Queries: We used three types of queries in our experiments: Single triple patterns queries (STP), star-shaped queries (S-1, S-2), and path-shaped queries (P-1, P-2, P-3). Single triple pattern (STP) queries consist of exactly one triple pattern in the query. Star-shaped and path-shaped queries are defined as in [9]. A S-k star-shaped query has one variable as subject and k joins, i.e., (k+1) triple patterns. An example of a S-1 star-shaped query is given in Figure 1. A P-k path-shaped query is generated by using the object of one triple pattern as subject in the next triple pattern, and it also contains (k+1) triple patterns. Previous work has shown that these query shapes are the most common shapes found in real-world RDF queries [17]. Our benchmark data consisted of 79 queries as shown in Table 3. Some query shapes could not be used on certain datasets due to the topology of the underlying ontology. For example, P-1 queries could not be sent to the Geo dataset since it only contained object properties. Each type a query was executed we used a random resource as subject or object, depending on the query type. The predicates of all queries are fixed.

Federated Query Engines: We implemented our DAW approach on top of three different federated query engines: DARQ [21], SPLENDID [8], and FedX [22]. Both DARQ and SPLENDID already provide an index with some of the statistics needed in DAW. Therefore, we only needed to extend this index. For FedX, which is index-free, we added an index similar to the one in DARQ with our DAW extension. The underlying query execution mechanism remained the same.

Metrics: We compared the three federated approaches against their DAW extensions. For each query type we measured (i) the average number of triple pattern-wise sources that were skipped, (ii) the average recall, and (iii) the average query execution time. We did not consider the number of endpoints requests, as it depends on a number of factors, such as join type, block and buffer size, that vary across the different federated query processors. The threshold was initially set to zero, in order to maximise recall while querying fewer sources. All experiments were carried out in a machine with a 2.53GHz i5 processor, 4 GB RAM, and 500 GB hard disk. Experiments were carried out in a local network, so the network costs were negligible. After the first warm up run, each query type was executed 10 times and results were averaged.

4.2 Experimental Results

Triple Pattern-Wise Source Skipping: Table 4 shows the number of capable triple pattern-wise sources that were skipped by our approach, for each query type, as well as the recall. The total number of triple pattern-wise sources selected by the original systems is shown in brackets. The threshold was set to zero, which means that only sources that were estimated to returned no new

Listing 1.2. A Single Triple Pattern (STP) query example

```
SELECT ?title WHERE
{ www2008-paper:103 pub:title ?title. }
```

Table 4. Distribution of the triple pattern-wise source skipped by DAW extensions for threshold value 0

Dataset	STP	S-1	S-2	P-1	P-2	P-3	Total	Recall
Diseasome	14(35)	30(77)	40(107)	35(65)	65(125)	30(50)	214(459)	100%
Geo	22(40)	23(55)	37(101)	-	-	-	82(196)	99.99%
Movie	22(38)	-	-	-	-	-	22(38)	100%
Publication	9(30)	10(37)	15(86)	14(60)	21(120)	32(102)	101(435)	100%
Total	67(143)	63(169)	92(294)	49(125)	86(245)	62(152)	419(1128)	-

(a) DARQ

Dataset	STP	S-1	S-2	P-1	P-2	P-3	Total	Recall
Diseasome	7(28)	30(77)	40(107)	35(65)	65(125)	30(50)	207(452)	100%
Geo	19(37)	23(55)	37(101)	-	-	-	79(193)	99.99%
Movie	15(31)	-	-	-	-	-	15(31)	100%
Publication	3(24)	10(37)	15(86)	14(60)	21(120)	32(102)	95(429)	100%
Total	44(120)	63(169)	92(294)	49(125)	86(245)	62(152)	396(1105)	-

(b) FedX and SPLENDID

results were pruned. We can see that DAW can effectively reduce the total triple pattern-wise selected sources, thus enable fewer subqueries federation. The highest gain was in the Diseasome dataset, where 214 sources were skipped in the DARQ approach, without affecting the recall. This corresponds to a decrease on the number of queried sources from 459 to 245. In other words, a full recall was achieved by querying only 53% of the available triple pattern-wise sources. In all cases except in the Geo dataset, the recall was not affected and all relevant results were retrieved. In the Geo dataset, the DAW index incorrectly pruned a small number of relevant sources, but the recall was still 99.99%. That means that DAW can deliver the same query results while querying much fewer sources. The source selection methods from FedX and SPLENDID return the same set of sources, therefore the number of skipped sources was the same for both. Moreover, they both use SPARQL ASK queries in the selection mechanisms, which leads to a better performance for STP queries. For example, consider the STP query given in Listing 1.2 where both the subject and predicate are bound. It is likely that a WWW2008 paper with id 103 is found in only one data source but the property `pub:title` may be found in every source. As a result, FedX and SPLENDID will only select a single capable source while DARQ will select all sources containing that predicate.

Query Execution Time: For each dataset and query type, we measured the average query execution time in each of the federated query approaches and also in their DAW extension. Again, the threshold was set to zero and the average was over 10 queries. Figures 3, 4, and 5 show the results. We can see

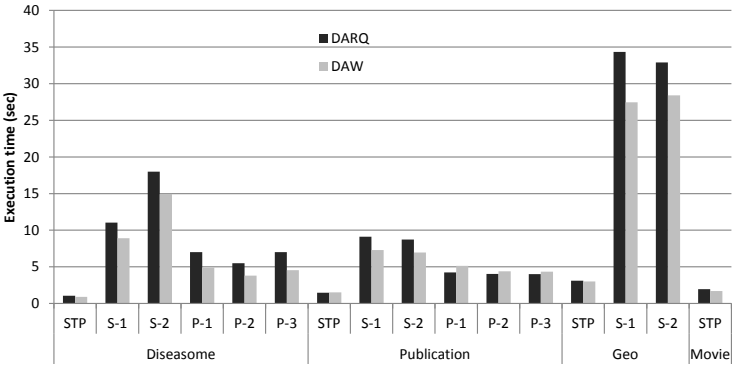


Fig. 3. Query execution time of DARQ and its DAW extension

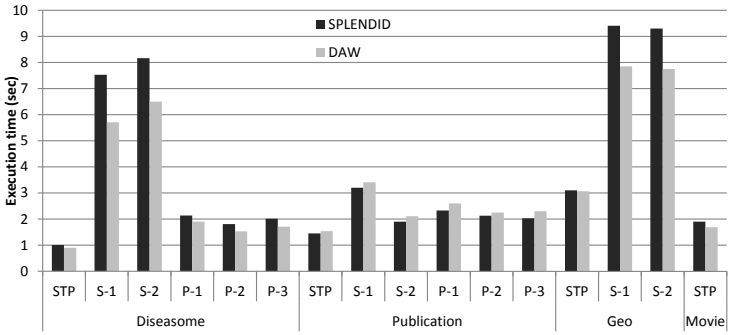


Fig. 4. Query execution time of SPLENDID and its DAW extension

that DAW improves the query performance for most of the cases. For three of the datasets, Diseasome, Geo and Movie, DAW improved the query execution times of all federated systems tested, for all query types. The query performance in the Diseasome dataset showed the highest improvements. This is due to the large number of triple pattern-wise sources that were pruned. We can also see that if the number of skipped sources is low – as for the Publication dataset – the overhead in computing the sources overlap can be higher than the execution time saved by querying fewer sources, so the overall query execution time is worse. The overall performance is summarised in Table 5. We were able to improve the query execution time in DARQ by 16.46%, the SPLENDID by 11.11%, and FedX by 9.76%. For the Diseasome dataset, the improvement for the DARQ approach was 23.34%. These are averaged values across all datasets and query types. DAW led to a performance gain for most of the settings. We expect that in a setup with larger datasets and higher overlap, DAW can lead to even better improvements.

Number of Queried Sources vs. Query Recall: The evaluation presented so far focused on achieving full recall, and only discarded sources that the DAW index estimated to contribute with no new results. We have shown that the

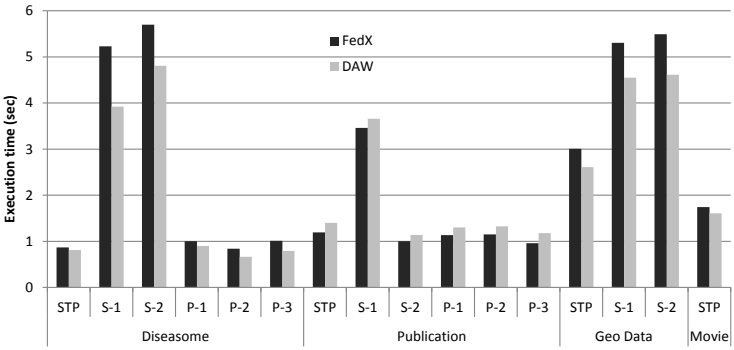
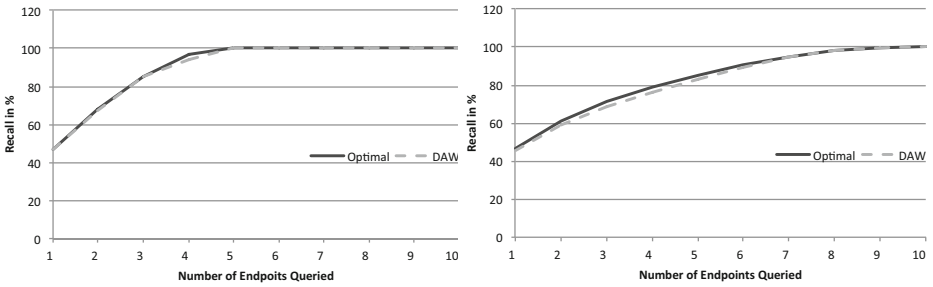


Fig. 5. Query execution time of FedX and its DAW extension

Table 5. Overall performance evaluation. *Exe.time* is the average execution time in seconds. *Gain* is the percentage in the performance improvement.

	Diseasome		Publication		Geo Data		Movie		Overall	
	Exe.time	Gain	Exe.time	Gain	Exe.time	Gain	Exe.time	Gain	Exe.time	Gain
DARQ	8.27		5.26		23.44		1.96		9.59	
DAW	6.34	23.34	4.94	6.14	19.62	16.31	1.68	13.88	8.01	16.46
SPLENDID	3.78		2.18		7.27		1.90		3.71	
DAW	3.04	19.48	2.38	-8.94	6.22	14.40	1.68	11.16	3.30	11.11
FedX	2.44		1.48		4.60		1.74		2.44	
DAW	1.98	18.79	1.67	-12.38	3.92	14.71	1.61	7.59	2.20	9.76

estimation given by our algorithm is quite accurate, as only 0.01% of the results in one dataset were missing. There might be cases, however, where full recall is not crucial and the query processing budget is limited. Here, the goal is to retrieve as many results as possible by querying only a subset of capable sources. Standard federated query processing approaches are only able to identify the set of capable sources. They are not able to compare the contribution of the sources in order to identify which subset yields to a better recall. With DAW, an approximation of this contribution is provided by the ranking step. For any given threshold, DAW is able to provide the subset of capable sources that will deliver the best recall for that number of sources. To demonstrate this, we computed the query recall for different threshold values for the DAW DARQ extension. We ran each of the STP queries 10 times on the Diseasome and Publication datasets and averaged the results. We varied the threshold value in order to limit the query to a fixed number of endpoints and we computed the query recall based on the DAW source selection. We compared it with the optimal duplicate-aware approach, where sources were manually selected to maximise the recall. The results are show in Figure 6. We can see that, in both cases, the source selection given by DAW is very close to the optimal case. Moreover, our experiment demonstrates the great potential in using source ranking for federated query processing. For the Diseasome dataset, by querying only 3 out of the 10 endpoints, DAW is able to retrieve 80% of the query results. A full recall is achieved with only 6



(a) Diseasome (b) Publication

Fig. 6. Recall for varied number of endpoints queried

endpoints. This naturally depends on the degree of overlap, but nevertheless it shows promising results that should be further explored.

5 Conclusion and Future Work

In this paper we presented DAW, an approach for duplicate-aware federated query over the Web of Data. DAW combines min-wise independent permutations with selectivity values to estimate the number of duplicate-free results. This estimation is used to first rank triple pattern-wise sources, based on their contribution, and to skip sources that contribute with little or no new results. DAW can be directly combined with existing index-assisted federated query processing systems, in order to improve the query execution. We evaluated our approach against DARQ, SPLENDID and FedX – three well known federated systems. The evaluation shows that by using the DAW extension the query execution times were improved in most of the cases, while recall was marginally affected. Moreover, DAW is suitable for maximising the recall for a fixed number of queried sources.

We will look at extending our index to further reduce the query execution time, for instance, by pre-computing some of the overlap statistics, based on query logs. The effect of different MIPs sizes and threshold values to find the optimal trade-off between execution time and recall will also explored, as well as different data partition methods.

Acknowledgments. This work has been supported by the European Commission under Contract No. FP720117287661 (GAMBAS), FP7-Granatum: RE7098, FP7-GeoKnow Grant No. 318159, by Science Foundation Ireland under Grant No. SFI/08/CE/I1380 (Lion-II) and Grant No. SFI/12/RC/2289 (INSIGHT).

References

1. Acosta, M., Vidal, M.-E., Lampo, T., Castillo, J., Ruckhaus, E.: Anapsid: an adaptive query processing engine for sparql endpoints. In: Aroyo, L., Welty, C., Alani, H., Taylor, J., Bernstein, A., Kagal, L., Noy, N., Blomqvist, E. (eds.) ISWC 2011, Part I. LNCS, vol. 7031, pp. 18–34. Springer, Heidelberg (2011)

2. Auer, S., Lehmann, J., Ngonga Ngomo, A.-C.: Introduction to linked data and its lifecycle on the web. In: Rudolph, S., Gottlob, G., Horrocks, I., van Harmelen, F. (eds.) Reasoning Weg 2013. LNCS, vol. 8067, pp. 1–90. Springer, Heidelberg (2013)
3. Basca, C., Bernstein, A.: Avalanche: putting the spirit of the web back into semantic web querying. In: SSWS, pp. 64–79 (November 2010)
4. Bender, M., Michel, S., Triantafillou, P., Weikum, G., Zimmer, C.: Improving collection selection with overlap awareness in p2p search engines. In: SIGIR, pp. 67–74 (2005)
5. Bloom, B.H.: Space/time trade-offs in hash coding with allowable errors. Commun. ACM 13(7), 422–426 (1970)
6. Broder, A.Z., Charikar, M., Frieze, A.M., Mitzenmacher, M.: Min-wise independent permutations. IJCSS 60, 327–336 (1998)
7. Drukh, N., Polyzotis, N., Garofalakis, M., Matias, Y.: Fractional xsketch synopses for xml databases. In: Bellahsène, Z., Milo, T., Rys, M., Suciu, D., Unland, R. (eds.) XSym 2004. LNCS, vol. 3186, pp. 189–203. Springer, Heidelberg (2004)
8. Görlitz, O., Staab, S.: Splendid: Sparql endpoint federation exploiting void descriptions. In: COLD, ISWC (2011)
9. Harth, A., Hose, K., Karnstedt, M., Polleres, A., Sattler, K.-U., Umbrich, J.: Data summaries for on-demand queries over linked data. In: WWW, pp. 411–420 (2010)
10. Hernandez, T., Kambhampati, S.: Improving text collection selection with coverage and overlap statistics. In: WWW (Special interest tracks and posters), pp. 1128–1129 (2005)
11. Hose, K., Schenkel, R.: Towards benefit-based rdf source selection for sparql queries. In: SWIM, p. 2 (2012)
12. Ladwig, G., Tran, T.: Linked data query processing strategies. In: Patel-Schneider, P.F., Pan, Y., Hitzler, P., Mika, P., Zhang, L., Pan, J.Z., Horrocks, I., Glimm, B. (eds.) ISWC 2010, Part I. LNCS, vol. 6496, pp. 453–469. Springer, Heidelberg (2010)
13. Langegger, A., Wöß, W., Blöchl, M.: A semantic web middleware for virtual data integration on the web. In: Bechhofer, S., Hauswirth, M., Hoffmann, J., Koubarakis, M. (eds.) ESWC 2008. LNCS, vol. 5021, pp. 493–507. Springer, Heidelberg (2008)
14. Li, Y., Heflin, J.: Using reformulation trees to optimize queries over distributed heterogeneous sources. In: Patel-Schneider, P.F., Pan, Y., Hitzler, P., Mika, P., Zhang, L., Pan, J.Z., Horrocks, I., Glimm, B. (eds.) ISWC 2010, Part I. LNCS, vol. 6496, pp. 502–517. Springer, Heidelberg (2010)
15. Michel, S., Bender, M., Triantafillou, P., Weikum, G.: IQN routing: Integrating quality and novelty in P2P querying and ranking. In: Ioannidis, Y., et al. (eds.) EDBT 2006. LNCS, vol. 3896, pp. 149–166. Springer, Heidelberg (2006)
16. Mitzenmacher, M.: Compressed bloom filters. IEEE/ACM Trans. Netw. 10(5), 604–612 (2002)
17. Morsey, M., Lehmann, J., Auer, S., Ngonga Ngomo, A.-C.: Dbpedia sparql benchmark: performance assessment with real queries on real data. In: Aroyo, L., Welty, C., Alani, H., Taylor, J., Bernstein, A., Kagal, L., Noy, N., Blomqvist, E. (eds.) ISWC 2011, Part I. LNCS, vol. 7031, pp. 454–469. Springer, Heidelberg (2011)
18. Nie, Z., Kambhampati, S., Hernandez, T.: Bibfinder/statminer: Effectively mining and using coverage and overlap statistics in data integration. In: VLDB, pp. 1097–1100 (2003)
19. Ntarmos, N., Triantafillou, P., Weikum, G.: Distributed hash sketches: Scalable, efficient, and accurate cardinality estimation for distributed multisets. ACM Trans. Comput. Syst., 27 (2009)
20. Polyzotis, N., Garofalakis, M.: Statistical synopses for graph-structured xml databases. In: SIGMOD, pp. 358–369 (2002)

21. Quilitz, B., Leser, U.: Querying distributed rdf data sources with sparql. In: Bechhofer, S., Hauswirth, M., Hoffmann, J., Koubarakis, M. (eds.) ESWC 2008. LNCS, vol. 5021, pp. 524–538. Springer, Heidelberg (2008)
22. Schwarte, A., Haase, P., Hose, K., Schenkel, R., Schmidt, M.: Fedx: Optimization techniques for federated query processing on linked data. In: Aroyo, L., Welty, C., Alani, H., Taylor, J., Bernstein, A., Kagal, L., Noy, N., Blomqvist, E. (eds.) ISWC 2011, Part I. LNCS, vol. 7031, pp. 601–616. Springer, Heidelberg (2011)
23. Shokouhi, M., Zobel, J.: Federated text retrieval from uncooperative overlapped collections. In: SIGIR, pp. 495–502 (2007)
24. Si, L., Callan, J.P.: Relevant document distribution estimation method for resource selection. In: SIGIR, pp. 298–305 (2003)

On the Status of Experimental Research on the Semantic Web

Heiner Stuckenschmidt, Michael Schuhmacher, Johannes Knopp,
Christian Meilicke, and Ansgar Scherp

Data - and Web Science Research Group,
University of Mannheim, Germany
firstname@informatik.uni-mannheim.de

Abstract. Experimentation is an important way to validate results of Semantic Web and Computer Science research in general. In this paper, we investigate the development and the current status of experimental work on the Semantic Web. Based on a corpus of 500 papers collected from the International Semantic Web Conferences (ISWC) over the past decade, we analyse the importance and the quality of experimental research conducted and compare it to general Computer Science. We observe that the amount and quality of experiments are steadily increasing over time. Unlike hypothesised, we cannot confirm a statistically significant correlation between a paper's citations and the amount of experimental work reported. Our analysis, however, shows that papers comparing themselves to other systems are more often cited than other papers.

1 Introduction

Popper characterizes the nature of science in terms of the falsifiability of claims [1]. Following this statement, careful validation of proposed methods and theories are commonly accepted as the core of reputable research. Over time different scientific disciplines have developed a variety of methodologies for evaluating results ranging from mathematical proofs to use cases and experiments. Semantic Web research and computer science as a whole is a discipline that has a strong formulative research approach [2]: it creates new formalisms, algorithms and systems claimed to be superior to previous proposals. If we follow the idea of reputable science, these claims have to be substantiated by a suitable method of validation, typically formal proofs, controlled experiments or use cases and examples. We claim that Semantic Web research is even more forced to validate scientific claims as it is a rather new area of research that often has to face prejudices of more established disciplines inside computer science and on the other hand faces the dilemma formulated by Wright: *"In [...] dynamic areas, researchers often face the choice: corroborating prior work to strengthen the foundations of the research area or 'pushing the envelope' while relying on prior work that may be less reliable"* [3]. We conclude that experimentation is an important way to validate results of Semantic Web research, especially as it has been argued that it challenges established results in more traditional disciplines [4] and is therefore less accessible to a strictly formal treatment.

H. Alani et al. (Eds.): ISWC 2013, Part I, LNCS 8218, pp. 591–606, 2013.

Having accepted that experimental research is important on the Semantic Web, we want to investigate the status of experimental research on the Semantic Web with respect to the quantity and the quality of experimental work. In particular, we want to compare the area of Semantic Web with other areas of computer science with respect to the importance given to experimental research. Further, we want to have a closer look at the way experiments are conducted to determine the usefulness of the reported experimental work for validating the claims and a reference for other researchers working on related problems. This question that is linked with the quality of the experimental work is much harder to capture than the pure amount of experimental work. Finally, we are interested in the question, whether doing experiments pays off in terms of research reputation and try to answer this question by analyzing citation statistics for papers with different amounts of experimental work.

Following the design of previous studies on experimental research in computer science (in particular [5] and [6]), we analyzed all papers from the International Semantic Web Conference starting in 2002 with respect to the type of work and amount of experimentation.

Going beyond previous studies, we also took a closer look at experiments with respect to the data used, the parameters measured, and the comparisons conducted.

This paper is structured as follows. In Section 2, we first give an overview of previous studies concerned with experimental work in computer science, summarizing the findings of these studies as a reference we can compare to. Subsequently, we define our research questions and hypotheses concerning the role of experimental work in Semantic Web research, provide more details about the data used, and the steps of the methodology that led us to our results (Section 3). This is followed by a detailed presentation and discussion of the results in Section 4. In Section 5, we conclude with discussing limitations of our study and the reliability of the results.

2 Empirical Studies of Experimental Research in Computer Science

Computer science is mostly regarded as a constructive science concerned with the creation of artefacts that cannot be entirely validated using formal methods [7]. Glass and others compare research approaches in different disciplines related to computer science [2]. Based on a review of major ACM and IEEE journals they conclude that almost 80% of computer science papers propose some new design or method that would actually require evaluation. While the amount of such papers is lower in certain subareas of computer science, like software engineering (55%), still a significant amount of work in computer science is formulative and requires some evaluation.

So far, the most detailed and systematic investigation of experimental research as a means for evaluating formulative research has been carried out by Tichy and others in 1995 [5]. Based on a sample of publications from major computer science journals the authors categorize papers into formal theory, design and modeling, as well as empirical work and others. The papers in the category design and modeling, which correspond to the formulative work in [2] are further analyzed with respect to the importance that is given to experimental work. For this purpose, Tichy and others further classified papers

in this category according to the space devoted to the description of experimental work. It turned out that in computer science literature experimental work is much less prominent than in engineering or natural sciences that were used as a reference. The study was repeated by Wainer and others focussing on a sample of papers published in 2005 by the ACM [6]. The authors used roughly the same setup and compared their results with the findings of Tichy and others, concluding that experimental work had gained importance, but is still behind the level found in other disciplines. We will discuss the results of these studies in more detail and compare them to our findings later.

Different additional studies have been performed in subdisciplines of computer science. Most notable in Software Engineering [8,9] and Computer-supported cooperative work [10,11]. Zelkowitz [8] identifies different forms of validation that can be found in the area of Software engineering and investigates the use of these different forms of validation in the Software Engineering literature in a quantitative study. In 2009 Zelkowitz repeated the study and reports the development over time [9]. He concludes that the amount of papers with a real evaluation has risen from only about 30% in 2000 to over 60% in 2009 moving towards the level that Tichy and others have presented for Computer Science as a whole. Pinelle and others [10] look at evaluation in papers on computer-supported collaborative work. The findings are in line with the above mentioned studies with a fraction of about 70% of the papers containing some kind of evaluation. On the other hand, only 30% of the papers used controlled experiments in a laboratory setting. Wainer and Barsottini performed a follow-up study on papers submitted to the ACM CSCW conference over a period of six years [11]. They found out that while overall the amount of experimental work has not increased, there was a significant increase in papers that performed an evaluation in terms of field experiments. Some smaller studies have been carried out in narrower fields. Prechelt performed a quantitative study of experimental approaches in the field of neural networks [12]. Like Machine Learning as a whole this area heavily depends on experimentation as a form of evaluation. Therefore the study is less concerned with the amount of experimentation, but with the specific setting of the experiments. As a central point of study, Prechelt looks at the nature and the number of datasets used in the experiments, discovering that most papers only use one single dataset as a basis for controlled experiments.

In summary, previous studies identified design as the dominant research methodology in Computer Science while empirical work is less important. Further, the studies showed that the importance of systematic experiments as a means of validating design research has gained importance over the last decades.

3 Research Questions and Method

The goal of this paper is to investigate the status of experimental research in the area of Semantic Web. In particular, we aim at investigating whether the importance of experimental work is comparable to the one in computer science in general as it has been identified in the previous studies discussed above. This question has two aspects: we need to identify work that can be characterized as Design and Modeling and therefore asks for an experimental evaluation. Having identified this work, we want to know whether experimental work has the same importance as in computer science in general.

As Semantic Web research is a rather young discipline, we are specifically interested in the development of the role of experimental research over time. Beyond these purely descriptive aspects, we also want to analyze the factors influencing the importance of experimental work. With respect to this, we look at the relation between amount and quality of experimental work and impact of a paper in terms of citations.

H1. Like in computer science in general, Design and Modeling work is the dominant form of research on the Semantic Web.
H2. The importance of experimental work on the Semantic Web is comparable with computer science in general.
H3. The importance of experimental work on the Semantic Web is increasing over time.
H4. The quality of experimental work on the Semantic Web is increasing over time.
H5. Strong experimental work increases the impact of a paper.

We conducted an empirical study for testing these hypotheses. For this purpose, we took the papers published at ISWC since 2002 and manually classified them according to the scheme proposed by [5]. In addition, we had a closer look at papers containing descriptions of experimental work with respect to the data used and the claims made. In the following, we describe the study design and the data used in detail and discuss the results of the study as well as the implications for the hypotheses stated above.

3.1 Data

As the goal of the study is to make valid assertions about the area of Semantic Web as a whole, the dataset used in the study has to be representative for the work conducted in the area. Making a good selection is complicated by the fact that the area of Semantic Web is not as well defined as more established research areas. Today, many conferences and journals contain work relevant for the Semantic Web. On the other hand, many researchers active in Semantic Web research also publish in other scientific disciplines such as artificial intelligence or database systems. Instead of trying to identify relevant work in different scientific outlets, we decided to focus on the International Semantic Web Conference as the major community event assuming that the work published there is representative for the whole area. Therefore, we included all full papers from the main research track of the ISWC conferences since 2002 instead of taking samples from different outlets. There are other potential sources of publications in particular, the ESWC and ASWC conference series as well as the Journal of Web Semantics and the Semantic Web Journal. Concerning ESWC and ASWC, we can safely assume that the ISWC conference series is the leading outlet and thus a more representative source of data. We explicitly decided against including journals, because conferences better reflect developments in young and dynamic fields such as the Semantic Web. The Journal of Web Semantics, however, might be included in future studies to compare the different kinds of publication outlets.

The dataset used in this study thus includes 500 papers from the following conference editions:

- 11. ISWC 2012: Boston, MA, USA (41 papers[1])
- 10. ISWC 2011: Bonn, Germany (50 papers[2])
- 9. ISWC 2010: Shanghai, China (51 papers[3])
- 8. ISWC 2009: Chantilly, VA, USA (43 papers, Research Track[4])
- 7. ISWC 2008: Karlsruhe, Germany (43 papers, Research Track[5])
- 6. ISWC / 2. ASWC 2007: Busan, Korea (50 papers, Research Track[6])
- 5. ISWC 2006: Athens, GA, USA (52 papers, Research Track[7])
- 4. ISWC 2005: Galway, Ireland (53 papers, Research Track[8])
- 3. ISWC 2004: Hiroshima, Japan (48 papers, Research Track[9])
- 2. ISWC 2003: Sanibel Island, Florida, USA (42 papers, Research Track[10])
- 1. ISWC 2002: Chia, Sardinia, Italy (27 papers, Research Track[11])

In order to measure the impact of papers in the dataset, we use citation statistics from Google Scholar (http://scholar.google.de/) and Microsoft Academic Search (http://academic.research.microsoft.com/). We use two different sources of citation statistics because it is well known that citation counts can differ significantly between different sources depending on the coverage of sources and the counting policy. Google Scholar has a very liberal counting policy that typically leads to a very high number of citations. In particular, as pointed out in [13], Google Scholar also covers grey literature citing a publication. Microsoft Academic Search is more conservative and counts fewer citations on average.

3.2 Annotation Scheme

In order to be able to compare our findings to previous studies on the role of experimental work in computer science as a whole, we used the classification scheme proposed in [5] with the modifications described in [11], i.e. the merge of the two categories 'Empirical Work' and 'Hypothesis Testing'. This allows us to relate our results to the finding reported in both papers. In particular, we classified papers according to the following four major categories.[12]

1) Formal Theory. Papers whose main contributions are formal propositions, e.g. lemmata and theorems and their proofs.

2) Design and Modeling. Papers whose main contributions are systems, techniques (e.g. algorithms) or models whose claimed properties cannot formally be proven.

[1] http://www.informatik.uni-trier.de/~ley/db/conf/semweb/iswc2012-1.html
[2] http://www.informatik.uni-trier.de/~ley/db/conf/semweb/iswc2011-1.html
[3] http://www.informatik.uni-trier.de/~ley/db/conf/semweb/iswc2010-1.html
[4] http://www.informatik.uni-trier.de/~ley/db/conf/semweb/iswc2009.html
[5] http://www.informatik.uni-trier.de/~ley/db/conf/semweb/iswc2008.html
[6] http://www.informatik.uni-trier.de/~ley/db/conf/semweb/iswc2007.html
[7] http://www.informatik.uni-trier.de/~ley/db/conf/semweb/iswc2006.html
[8] http://www.informatik.uni-trier.de/~ley/db/conf/semweb/iswc2005.html
[9] http://www.informatik.uni-trier.de/~ley/db/conf/semweb/iswc2004.html
[10] http://www.informatik.uni-trier.de/~ley/db/conf/semweb/iswc2003.html
[11] http://www.informatik.uni-trier.de/~ley/db/conf/semweb/iswc2002.html
[12] Descriptions of categories are taken from [5].

3) Empirical Work / Hypothesis Testing. Papers that collect, analyze and interpret observations about known designs, systems, models or hypotheses.
4) Other. Papers that do not fit the other categories (e.g. surveys).

Further, we annotated all papers in Category 2 with additional information about the experiments conducted. Following Tichy et al., we use the number of pages devoted to the description of the experiment and its outcome as an indicator for importance of the experimental work and therefore annotate every paper with the number of pages describing experiments and the fraction of the overall paper they constitute.

Further, we annotate all papers of Category 2 with the following information about the nature of the experiments.

Standard Used for Comparison. Does the paper report about *different settings* or the system or method? Are results compared against *existing baselines*? Are results compared against the results of *other systems*? The latter includes both indirect comparisons against results reported in other papers and direct comparisons obtained by executing the other system as part of the experiments.
Datasets Used. Has *one dataset* been used or have *several datasets* been used within the experiments? Has the dataset been *self-created* by the authors for the purpose of conducting the experiments or is it *externally provided*?

We use this information as an indication of the quality of the experimental design, assuming that an ideal experimental design will compare a proposed system against other leading systems or at least sensible baselines using several datasets with different characteristics. One can argue about whether externally provided datasets should be preferred over self-created ones, in many cases externally provided datasets are publicly accessible benchmarks that support the comparison with other systems, which we consider desirable.

3.3 Study Design

Annotation Process. The classification of papers into the four categories was performed manually by a group of five annotators, three of which were senior and two junior level researchers. One of the senior researchers acted as a judge, while the other four were annotators. We started with the 2012 papers which were annotated by all four annotators to get a feeling for the level of agreement and discuss difficult cases to reach a common understanding of the category definitions and typical problems. In a second round, the remaining papers were annotated by two groups consisting of one senior and one junior annotator. One group performed the annotation of papers from even years, the other of papers from odd years. All papers where the annotators disagreed on the correct category were forwarded to the judge who made a final decision on the category.

In the same way, the number of pages devoted to experimental work was annotated at a granularity of half pages. First all annotators determined the number of pages for the 2012 conference. Subsequently, all remaining papers were annotated in two groups.

Papers with a disagreement were forwarded to the judge. In case of consensus on the category and a disagreement of just half a page, the judgment of the senior annotator was used. The detailed analysis of the experimental setting was carried out by two senior researchers where one annotated the papers from odd and another annotated the papers from even years.

In order to check how hard it is to decide on the classification of each paper, the inter-annotator agreement for both annotator pairs in the second round was computed using Cohen's Kappa [14]. The result for each pair is a number between zero and one, where zero means that the agreement between both annotators cannot be distinguished from chance, while one means perfect agreement. The annotators of the odd years reached a kappa of $\kappa = 0.63$ while the group annotating the even years scored $\kappa = 0.47$. There is no universally accepted value range defined for Cohen's Kappa, but there are interpretations of Cohen's Kappa in the literature that say these results can be considered to be *moderate* (even years) and *substantial* (odd years) agreement [15]. It is safe to say that the kappa values easily exceed an agreement by chance which means that the classification task was well defined. Most disagreement result from confusions between Category 1 and 2, i.e. 32 out of 77 disagreements. That means it is often unclear whether a paper should be considered as a theoretical paper or a modeling paper without experiments. All disagreements were finally resolved by the decision of the judge.

Test of Hypotheses. Based on the classification of papers according to the four main categories, we compare the distribution of papers from ISWC to the distributions reported in previous studies for general computer science and other disciplines (H1). Further, we look at papers from Category 2 (Design and Modeling) in more detail. In particular, we analyze how the papers distribute across the subcategories defined by the fraction of the pages devoted to the description of experimental work (0%, (0% - 10%], (20% - 50%], $>$ 50%) and compare the distribution with previous studies (H2). We then look at the development of experimental work over time by plotting the distribution of papers across all categories over the past eleven years. We also look at the average number of pages devoted to experimental work in the different years and compute the correlation between year of publication and number of pages (H3). In a similar way we look at the experimental setting in more detail. For papers from Category 2 we analyze the standards used for comparisons and the datasets used as input to the experiments. We interpret these features and their characteristics as indicators for experimental quality in terms of significance and validity and analyze whether the experimental quality has increased over the past eleven years (H4). Finally, we used statistical models to test for correlation between the pages devoted to experimental work and the features that are indicators for experimental quality on the one hand and the impact of the paper on the other hand. We control the influence of other variables, such as the year of publication, to avoid spurious correlations, that do not appropriately reflect the dependencies between experimental work and its influence on a papers impact (H5).

4 Results

In the following, we discuss our findings regarding the different hypotheses in more detail. In particular, we present descriptive statistics of the ISWC paper collection and results of investigating possible correlations with research impact.

4.1 H1. Like in Computer Science in General Design and Modeling Work Is the Dominant Form of Research on the Semantic Web

We investigated the first hypothesis by comparing the distribution of papers across the four main categories 'Formal Theory', 'Design and Modeling', 'Empirical Work' and 'Other', with the results of the previous studies conducted by Tichy et al. and Wainer et al. respectively. The results are shown in Table 1.

Table 1. Comparison of the relative share of papers in each of the four research method categories. While the figures from [5] refer to papers published in 1995, and [6] used papers from 2005, our study covers 11 consecutive years from 2002 – 2012. This also explains the comparably high number of papers (500) included in our study. Noteworthy is that all three studies found a similar pattern, revealing Category 2) Design and Modeling as being the domination method of research.

	ISWC 2002-2012	[6]	[5]
1) Formal Theory	11.2% (56)	4.1% (6)	18.7% (48)
2) Design / Modeling	**80.8% (404)**	**70.1% (103)**	**64.1% (164)**
3) Empirical Work	5.4% (27)	22.4% (33)	10.2% (26)
4) Other	2.3% (13)	3.4% (5)	7.0% (18)
	100% (500)	100% (147)	100% (256)

Looking at the results, we see that like in previous studies, most of the work, namely 80.8% falls into the category 'Design and Modeling' while 11.2% of the work is of theoretical nature and only 5.4% is empirical work in the sense of our classification, leaving 2.3% other papers. This confirms our hypothesis that Design and Modeling is the dominant form of research on the Semantic Web. Comparing this to the results of the previous study, we can see that the dominance of design and modeling work is even more visible than in the previous studies, where 64.1% and 70.1% of the work was classified as Design and Modeling. Partially this difference can be explained by the general trend to more practical work in computer science and the different periods the studies were carried out: While Tichy and others only considered papers published in 1993 and Wainer and others analyzed papers published in 2005, our study includes papers published between 2002 and 2012. This means that our results should at least be comparable with the results from Wainer and others that fall into the period covered by our study.

Another noticeable observation is the lack of a significant amount of empirical work on the Semantic Web. With only 5.4% of all papers, the fraction of empirical work is only half as large as in the 1995 study and only a quarter of the amount found by the 2005 study. In fact, besides some papers that investigated the amount and nature of linked data and ontologies found on the web, there is no empirical work concerned with Semantic Web technologies. This could be explained by the fact that the Semantic Web is still a very young area of research where the focus is still on creating new technologies rather than on analyzing the impact of the new technologies on the Web.

4.2 H2. The Importance of Experimental Work on the Semantic Web Is Comparable with Computer Science in General

We investigate the claim that experimental work has the same importance in the Semantic Web area as in Computer Science in General based on the criterion of importance proposed by Tichy and others in their original study. In particular, Tichy and others propose to use the fraction of the paper devoted to the description of experimental work. We follow this suggestion and compare the distribution of papers in the relevant category of Design and Modeling Papers across the different subcategories proposed by Tichy and others.

Fig. 1. Comparison between three studies reporting on the relative share of pages of Category 2 papers dedicated to experiments. While the figures from [5] refer to papers published in 1995, and [6] used papers from 2005, our study covers 11 consecutive years from 2002 – 2012.

Figure 1 compares the distribution of papers across classes between our study and the two previous studies looking at Computer Science in general. The first observation is that in the study of Tichy conducted in 1995 the fraction of Design and Modeling papers that contained no description of experimental work at all is significantly larger (45% vs. 31% and 33%) while the fraction of papers with more than 20% of the pages devoted to the description of experiments is significantly smaller (approx. 26% vs. 34% and 36%) than in the other two studies. This visible difference, again can be explained by the general increase of importance of experimental work since the early Nineties. Comparing our results to the Study of Wainer et al., we can see that the difference between the distribution is very small. Except for the category of papers with more than 50% of the pages devoted to experimental work, the differences between the classes are always within two percentage points. This seems to suggest that the importance of experimental work on the Semantic Web is comparable with General Computer Science literature published by the ACM.

Being aware of the general tendency that experiments become more important over time, we take another look at the papers from the study of Wainer and others and the papers from ISWC 2005 to be able to directly compare papers published in the same year. The results are summarized in Table 2.

Table 2. Comparison of the relative share of pages of Category 2 papers dedicated to experiments. For both studies, ours and [6], we report only papers from 2005 here. We observe a generally lower amount of pages for experiments when comparing ISWC to general Computer Science.

	ISWC 2005	ACM Sample 2005 [6]
0%	40%	33%
(0% - 10%]	11.1%	9.7%
(10% - 20%]	15.6%	21.4%
(20% - 50%]	28.9%	30.1%
$> 50\%$	4.4%	5.8%

Here, we observe a slightly different picture. When only looking at papers from 2005, we see that the fraction of papers without any experimentation is higher (40%) than the figure reported by Wainer and others (33%) and also higher than the average fraction across all ISWC conferences. On the other hand, the fraction of papers with more than 10% of the pages describing experiments is lower (49%) compared to the study by Wainer (57%) and also much lower than the average across all ISWC conference (also 57%). We conclude that at least in 2005, experimental work did not yet have the same level of importance in Semantic Web research than in general Computer Science, while averaged across all ISWC conferences, the importance is comparable to general Computer Science in 2005.

4.3 H3. The Importance of Experimental Work on the Semantic Web Is Increasing over Time

The inconclusive result of comparing the number of pages as an indicator for the importance of experimental work across the different studies asks for a deeper analysis of the development of the indicator over time. We explain the observation that, while in 2005 experimentation was not as prominent in ISWC papers than in general computer science, the results measured across all ISWC conferences was comparable with the results of the 2005 study by Wainer et al. by hypothesizing that the importance of experimental work was rather low in the early years of the ISWC conference. This is not uncommon for new fields of research, as first, the principled ideas have to be laid out and basic ideas have to be tested in prototypical form. Only later, when the field is more established and the problems are better understood, systematic experiments become the standard way of validation. As the first ISWC conference took place in 2002, the field was still in a rather early stage in 2005. According to our hypotheses H3, we expect the importance to have significantly increased since then, which would explain the result over all conferences.

We test this hypotheses by looking at the development of the different categories over the years 2002 to 2012. In particular, we look at the development of the different subcategories under Design and Modeling to get an impression, whether the importance of experiments is increasing in this category. The results are summarized in Figure 2. The first observation to be made is, that the overall amount of papers in Design and Modeling stays roughly the same - around 80% - with a slight decrease to about 75% in the last two years. Inside this category, however, we can see a radical shift in the classification from 2002 to 2012. The shift can best be observed when looking at the

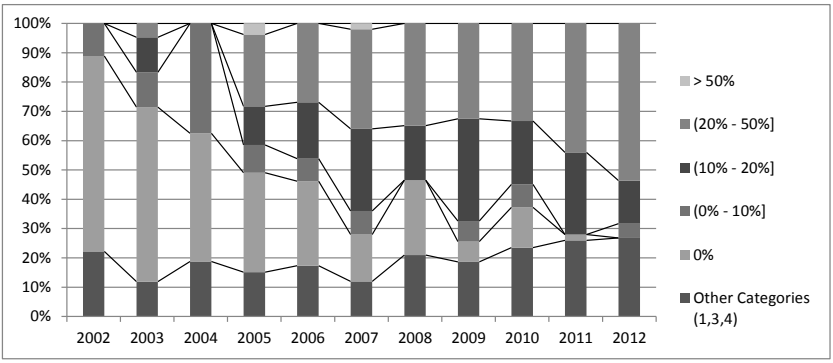


Fig. 2. Barchart showing the relative share of papers of Category 2 (Design and Modeling), grouped by the relative number of pages dedicated to experiments per year: 0%, $(0\% - 10\%]$, $(10\% - 20\%]$, $(20\% - 50\%]$, $> 50\%$. All other categories $(1, 3, 4)$ are summarized in one class. Most noteworthy is the decrease over time for papers without any experiments (0% pages), while the group of $(20\% - 50\%]$ is growing.

subcategory of papers with 0% of pages describing experimentation and the subcategory of papers with 20% to 50% of the pages devoted to experimentation. While the former category contained about 70% of the papers in 2002 it completely disappeared by 2012, showing that today Design and Modeling papers without experimentations are not any more considered to be adequate. On the other hand, the amount of papers with 20-50% experimentation show a constant increase and represents more than 50% of the papers in 2012. In 2005 there were still more papers without experiments (about 35%) than papers with 20-50% (about 25%), which explains the results reported above.

The increase in importance can also be observed well when directly looking at the number of pages instead of the categories. Figure 3 shows a standard box-plot for the relative number of experiment pages for Category 2 (Design and Modeling) papers. We identify a trend of growing importance of experiments over time. With the exception of 2010, the median is constantly rising up to 25% in 2012. Measuring this trend in figures, the Spearman Correlation Coefficient is statistically significant ($r_S(402) = .49, p < .000$).

4.4 H4. The Quality of Experimental Work on the Semantic Web Is Increasing over Time

With respect to H4, we decided to focus on four binary variables as indicators for experimental quality. Our choice is based on the following assumptions.

- Using several datasets is better than using only one dataset (SEVERAL).
- Using an already existing dataset is better than using a dataset that has been created for the purpose of conducting the experiments (OTHER).[13]

[13] During our annotation phase, we observed problems in deciding whether a simple setting should or should not be treated as a baseline. For that reason we did not distinguish between comparisons against a baseline and comparisons of different settings and counted each such comparison in the same variable.

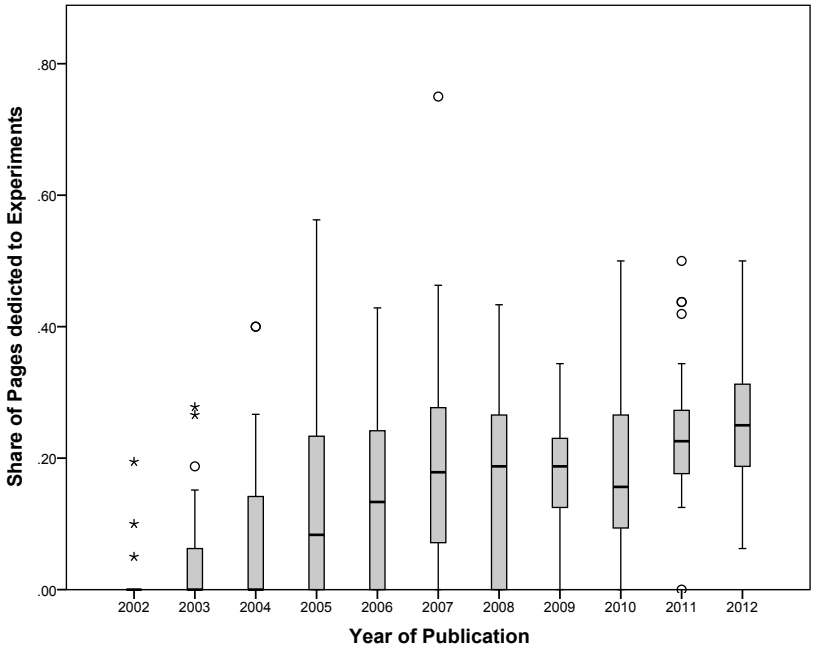


Fig. 3. Box-plot showing the relative number of pages of Category 2 (Design and Modeling) papers by year of publication. The median starting at 0% in 2002 increases constantly (with the exception of 2010) over time, reaching its top of 24% in 2012. The second/third quartile, denoted by the box, varies, but is since 2009 clearly above zero. Outliers are displayed as circles/stars.

- Comparing the proposed approach against a baseline or comparing different settings against each other is better than no such comparison (BASEDIFF).
- Comparing the proposed approach against other algorithms/systems is better than no comparison (SYS).

The variables SEVERAL and OTHER can be interpreted as indicators for the universal validity of the reported results. The variables BASEDIFF and SYS indicate whether the authors informed the reader on the performance (e.g. runtimes), quality (e.g. precision), or usability compared to alternative approaches. Without such a comparison, it is hardly possible to draw any conclusions related to the improvements made.

The results of our analysis are shown in Figure 4, where we depicted the countings for all four variables with respect to Category 2 papers. Figure 4 reveals a clear trend. The quality of experimental work is increasing over time with respect to each variable. In 2003 only a minor share of all papers had a positive characteristic in one of the four variables, while in 2012 more than 50% of all papers had a positive characteristic in three of four variables. However, only 33% of all papers in 2012 compared their results against other systems (SYS). While this is an improvement compared to the previous years, there are still many papers that do not compare their results against other systems. We computed also the correlation between the year of publication and the four quality measures using Spearman's rank correlation coefficient. We find that all

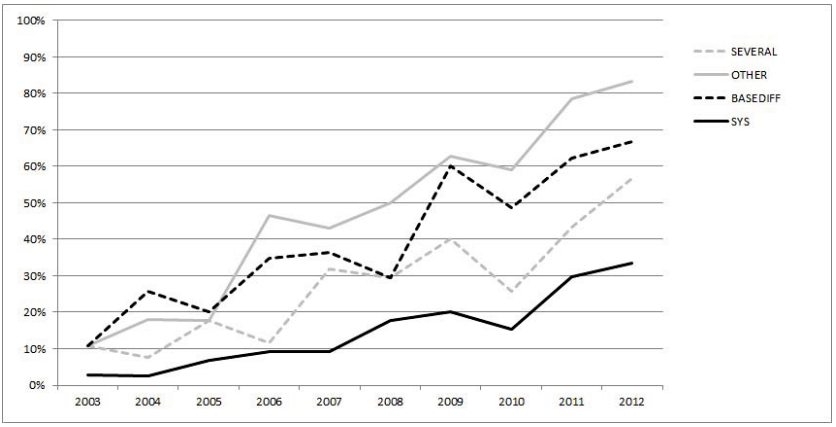


Fig. 4. Development of relative share of Category 2 (Design and Modeling) papers complying to different evaluation quality indicators over time. While all indicators start at a low level of $\leq 11\%$ in 2003 and rise with the years, we found the usage of externally provided datasets (OTHER) to increase the most. Nevertheless, even in 2012 only about on third of all papers compare themselves to other existing systems (SYS).

variables show positive and statistically significant correlations with the year of publication (r_S between .36 and .46, $p \leq .000$).

Our observations can be explained by two factors. One factor might be an increasing awareness of the importance attributed to experimental work. Another factor might be the general development of the community. What has been a novel area of research 10 years ago, might have become an established research area associated with well-defined problems, commonly accepted formats, well-known datasets and accepted benchmarks. Obviously, both factors go hand in hand, resulting in the positive trend that we reported in our evaluation.

4.5 H5. Strong Experimental Work Increases the Impact of a Paper

For analyzing the potential relation between the amount of experimental work of a paper and its impact, we employ a generalized linear model (GLM). As described above, we take the relative number of pages describing experiments (RELPAGES) as a general proxy for the importance of the experimental part within a paper. Following [5], we thus make the assumption that the better the experiments in a paper are, the more the paper reports about the experiments.

For measuring the impact of a paper, we divide the citation count by the age of a paper in years (RELCITATIONPA). We use only Google Scholar data, as the Microsoft Academics citation figures are strongly correlated with the Google Scholar data, the statistically significant Pearson's correlation is 0.969, and would thus result in similar findings.

Our first analysis reviews the simple pairwise variable correlation. For all correlations here $df = 402$. We find that a statistically significant Spearman's correlation

Table 3. Spearman's Correlation Coefficient for the citation count per year, the relative number of pages for experiments, and the year of publication. All correlations are statistically significant and negative with respect to the year of publication. The experimental pages count was surveyed only for the 404 Category 2 papers.

	Correlation	Sig. (2-tailed)	N
Scholar Citation Count per Year (RELCITATIONPA)	1.000	.000	500
Experimental Pages Count (RELPAGES)	-0.175	.000	404
Year of publication (YEAR)	-0.458	.000	500

($r_S = -0.175$) between RELCITATIONPA and RELPAGES exists. This would indicate that a decrease in the number of citations goes hand in hand with an increase of the amount of pages spend on experiments. But as shown in Table 3, we also observe a strong correlation of -0.458 between RELCITATIONPA and the year of publication (YEAR). This is consistent with the temporal development reported above in Section 4.3. Thus, from this data it cannot be concluded if RELCITATIONPA effects the impact of a paper, or if both developments just coincided with the general development over time. For a more fine-grained analysis of the effects, we thus use a GLM for regression analysis.

The GLM takes a log-link function to explain the outcome of our dependent variable RELCITATIONPA as the result of a $Tweedie(p = 1.5)$-Distribution[14] taking into account RELPAGES and AGE, as well as the binary quality measures BASEDIFF, SYS, OTHER, and SEVERAL. We choose a Tweedie distribution function as it suits citation count data, where several papers have zero citations, well. In addition, we find that the model shows better goodness of fit values, measured by the Akaike Information Criterion (AIC), for our data, compared to a linear regression as well as to a loglinear GLM with a Poisson distribution (Poisson-Regression). The Omnibus likelihood-ratio Chi-Square test of our model ($df = 18$, cf. Table 4) versus the intercept-only model confirms a significant difference ($p \leq .000$). We use the 404 Category 2 (Design and Modeling) papers and group the values for RELPAGES into the five classes described above, see e.g. Figure 2, and denoted them in the following by the variable RELPAGESCLASS.

Under this model, RELPAGESCLASS has a negative parameter and thus indicates a negative effect of the number of pages on the likelihood to be cited. But as the model effects test given in Table 4 reports a non-significance ($p = .239$), we cannot conclude that RELPAGESCLASS has an effect on RELCITATIONPA. On the other hand, SYS and SEVERAL both have a statistical significance ($p = .050$ and $p = .053$). SEVERAL has a positive parameter (0.250), thus indicating a positive effect of reporting experimental results for different datasets and not only for one dataset. We make a similar observation for papers comparing their own system to other systems, as SYS has also a positive parameter (0.301). Because both variables, SYS and SEVERAL, are strong indicators for profound experimental work, we may take this finding as a support for our hypothesis and conclude, that comparing oneself to others increases the likelihood to get cited. Nevertheless, if this correlations reveals a causality remains somehow doubtful, as the positive correlation may also originate from the fact that papers with an active and

[14] For $1 < p < 2$, this is a compound Poisson-Gamma distribution with a point mass at zero.

Table 4. Test of Model Effects with Wald-Chi-Square for the GLM with RELCITATIONPA as depended variable. Variable RELPAGESCLASS is not statistically significant for the model, but the two quality indicators SYS and SEVERAL have significance test values of $p \leq .05$ and .053.

	Wald Chi-Square	Deg. of Freedom	Significance
(Intercept)	281.760	1	.000
RELPAGESCLASS	5.510	4	.239
AGE	164.634	10	.000
BASEDIFF	3.139	1	.076
SYS	3.835	1	.050
OTHER	0.205	1	.651
SEVERAL	3.750	1	.053

large research community have more opportunities to cite other systems, while papers on isolated topics simply do not have peer papers to related to. Regarding all other variables, our model can currently not give a statistically reliable explanation whether they have any effect or not.

5 Conclusion

After more than 10 years of Semantic Web conferences, we believe it has been time to conduct a study like this. It serves a basis for a backward analysis of what has happened so far alike as actuates fruitful future discussions that will help steering the kind of research conducted in our community. Our main aim was to learn how the field of Semantic Web research is doing compared to general computer science and to show that the field is on its way to become an established scientific discipline with high standards concerning experimental evaluation of work.

Our results confirm that Semantic Web, as other emerging fields, has undergone a significant change with respect to the importance and quality of experimental work. We found that the amount of experimental work done is comparable to Computer Science in general and that the quality of experiments in terms of the use of publicly available datasets and comparison to other systems and benchmarks has continuously increased over the last ten years. In particular, we see that today it is virtually impossible to get design and modeling work accepted in the main track of ISWC without having experimental results. Further, our results show that papers that relate their contribution to existing datasets or other systems are more often cited than others.

As next steps, we will add more Semantic Web conferences like ESWC and journals such as Journal of Web Semantics and Semantic Web Journal. In addition, we will conduct more detailed analyses such as investigating the influence of (co-)authorship with respect to citations. We also plan to conduct analyses of the citations based on the single ISWC conferences. To this end, we are looking at the papers published in a specific year of a conference only and investigate whether the papers of a specific category are statistically more cited than papers in other categories. Finally, we would like to automatically obtain the topic of the articles by extracting topic models from the abstracts and analysing the citation distribution over these topics.

References

1. Popper, K.: The logic of scientific discovery. Routledge (2002)
2. Glass, R., Ramesh, V., Vessey, I.: An analysis of research in computing disciplines. Communications of the ACM 47, 89–94 (2004)
3. Wright, D.: Motivation, design, and ubiquity: A discussion of research ethics and computer science. arXiv preprint arXiv:0706.0484 (2007)
4. van Harmelen, F.: Where Does It Break? or: Why the Semantic Web Is Not Just "Research as Usual". In: Sure, Y., Domingue, J. (eds.) ESWC 2006. LNCS, vol. 4011, p. 1. Springer, Heidelberg (2006)
5. Tichy, W., Lukowicz, P., Prechelt, L., Heinz, E.: Experimental evaluation in computer science: A quantitative study. Journal of Systems and Software 28, 9–18 (1995)
6. Wainer, J., Novoa Barsottini, C., Lacerda, D., Magalhães de Marco, L.: Empirical evaluation in computer science research published by acm. Information and Software Technology 51, 1081–1085 (2009)
7. Ramesh, V., Glass, R., Vessey, I.: Research in computer science: an empirical study. Journal of Systems and Software 70, 165–176 (2004)
8. Zelkowitz, M., Wallace, D.: Experimental models for validating technology. Computer 31, 23–31 (1998)
9. Zelkowitz, M.: An update to experimental models for validating computer technology. Journal of Systems and Software 82, 373–376 (2009)
10. Pinelle, D., Gutwin, C.: A review of groupware evaluations. In: Proeedings of the IEEE 9th International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WET ICE 2000), pp. 86–91. IEEE (2000)
11. Wainer, J., Barsottini, C.: Empirical research in CSCW a review of the ACM/CSCW conferences from 1998 to 2004. Journal of the Brazilian Computer Society 13, 27–36 (2007)
12. Prechelt, L.: A quantitative study of experimental evaluations of neural network learning algorithms: Current research practice. Neural Networks 9, 457–462 (1996)
13. Bauer, K., Bakkalbasi, N.: An examination of citation counts in a new scholarly communication environment. D-Lib Magazine (2005)
14. Cohen, J., et al.: A coefficient of agreement for nominal scales. Educational and Psychological Measurement 20, 37–46 (1960)
15. Viera, A.J., Garrett, J.M.: Understanding interobserver agreement: The kappa statistic. Fam. Med. 37, 360–363 (2005)

A Graph-Based Approach to Learn Semantic Descriptions of Data Sources

Mohsen Taheriyan, Craig A. Knoblock, Pedro Szekely, and José Luis Ambite

University of Southern California
Information Sciences Institute and Department of Computer Science
{mohsen,knoblock,pszekely,ambite}@isi.edu

Abstract. Semantic models of data sources and services provide support to automate many tasks such as source discovery, data integration, and service composition, but writing these semantic descriptions by hand is a tedious and time-consuming task. Most of the related work focuses on automatic annotation with classes or properties of source attributes or input and output parameters. However, constructing a source model that includes the relationships between the attributes in addition to their semantic types remains a largely unsolved problem. In this paper, we present a graph-based approach to hypothesize a rich semantic description of a new target source from a set of known sources that have been modeled over the same domain ontology. We exploit the domain ontology and the known source models to build a graph that represents the space of plausible source descriptions. Then, we compute the top k candidates and suggest to the user a ranked list of the semantic models for the new source. The approach takes into account user corrections to learn more accurate semantic descriptions of future data sources. Our evaluation shows that our method produces models that are twice as accurate than the models produced using a state of the art system that does not learn from prior models.

Keywords: semantic description, source modeling, source description, semantic model, Semantic Web.

1 Introduction

Today, information sources such as relational databases and Web services provide a vast amount of structured data. A common approach to integrate sources involves building a global model and constructing source descriptions that specify mappings between the sources and the global model [8]. In the traditional data integration approaches, source descriptions are specified as global-as-view (GAV) or local-as-view (LAV) descriptions. In the Semantic Web, what is meant by a source description is a semantic model describing the source in terms of the concepts and relationships defined by an ontology. This semantic model can be viewed as a graph with ontology classes as the nodes and ontology properties as the links between the nodes.

H. Alani et al. (Eds.): ISWC 2013, Part I, LNCS 8218, pp. 607–623, 2013.

The first step in building a source description of a source is to determine the semantic types. That is, each source attribute is labeled with a class or a data property of the domain ontology. However, simply annotating the attributes is not sufficient. For example, consider a data table with two columns: *name*, which is mapped to the class *Person*, and *place*, which is mapped to the class *City*. Unless the relationship between the two columns is explicitly specified, we do not know whether the city is the birth place of the person or it is the place where she currently lives. A precise source description needs a second step that determines the relationships between attributes in terms of properties in the ontology.

Writing source descriptions by hand requires significant effort and expertise. Although desirable, generating these descriptions automatically is a challenging problem [1, 7, 10, 17, 20]. Most of the proposed approaches on the Semantic Web focus on the first step of the modeling process. In our previous work [13], we presented an algorithm to construct semantic models of data sources by computing a Steiner tree in a graph derived from the ontology and the semantic types. If the suggested tree is not the correct model of the data, the user interactively imposes constraints on the algorithm through a graphical user interface to build the correct model. However, the system does not learn the refinements done by the user and always suggests a random minimal tree as the initial model of the new sources.

In this work, we present algorithms to improve the quality of the automatically generated models by using the already modeled sources to learn the patterns that more likely represent the intended meaning of a new source. The insight of our approach is that different sources in the same domain often provide similar or overlapping data. Thus, it should be possible to exploit knowledge of previously modeled sources to learn descriptions for new sources. First, we construct a graph whose main components are subgraphs corresponding to the known source models. We use the domain ontology to infer the possible paths connecting the nodes across different subgraphs. This graph models the space of plausible source descriptions. Next, we label each source attribute with a semantic type and try to find a set of candidate mappings between these semantic types and the nodes in the graph. For each resulting set of the nodes, we compute the minimal tree that connects them and consider this tree as a plausible source model. Then, we score the models to prefer the ones formed with more coherent and frequent patterns. Finally, we generate a ranked list of possible semantic models. We can put users in the loop by allowing them to select the correct model or refine one of the suggested models. The correct model will be added to the graph as a new component yielding more accurate models in the future.

Our work provides a basis to learn the source descriptions of information sources. The main contribution of our work are the techniques to leverage attribute relationships in known source descriptions to hypothesize attribute relationships for new sources, and capturing them in source descriptions. Such source descriptions are beneficial to automate tasks such as source discovery, information integration, and service composition. They also make it possible to convert sources into RDF and publish them in the Linked Data cloud [23].

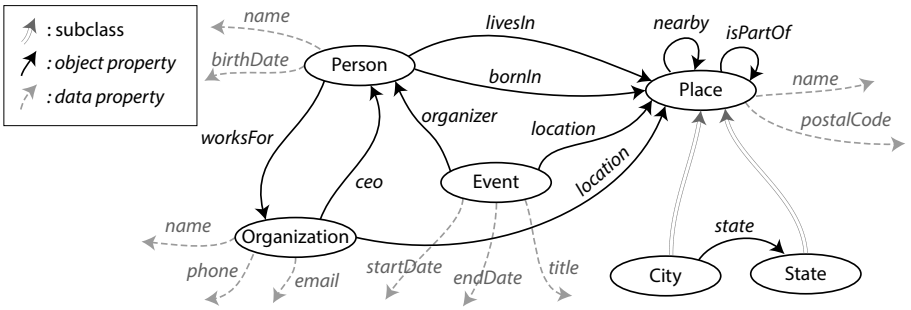


Fig. 1. The ontology that we use to model the sources in the example

2 Motivating Example

In this section, we explain the problem of learning source descriptions by giving a concrete example that will be used throughout the paper to illustrate our approach. In this example we have five data sources whose definitions are as follows:

$s_1 = personalInfo(name, birthdate, city, state, workplace)$
$s_2 = getCities(state, city)$
$s_3 = businessInfo(company, ceo, city, state)$
$s_4 = getEmployees(employer, employee)$
$s_5 = postalCodeLookup(zipcode, city, state)$

s_1 is a dataset providing information about people; s_2 is a Web service that takes as input a U.S. state and returns all the cities of that state; s_3 is a dataset providing information about businesses such as their name; s_4 is a list of U.S. companies and their employees; and s_5 is a Web service that returns the list of all the cities in a ZIP code. We use the ontology shown in Figure 1 to build a source description for each source. These descriptions are illustrated in Figure 2. Now, suppose that the first three sources (s_1, s_2, and s_3) have already been modeled and the other two (s_4 and s_5) are new sources not modeled yet. The goal is to automatically infer the source descriptions for s_4 and s_5 given the ontology and the models for s_1, s_2, and s_3.

Automatically building a source description of an unknown source is difficult. First, the system must map the source attributes to classes in the ontology. Considering source s_4 in Figure 2, attribute *employer* should be mapped to *name* of *Organization* and attribute *employee* should be mapped to *name* of *Person*. Next, the system needs to infer the relationships between the classes used to model the attributes. Our sample ontology has two links between *Person* and *Organization*, namely *worksFor* and *ceo*. The system needs to select *worksFor*, which captures the intended meaning of s_4. The problem is more complicated in cases when the relevant classes are not directly connected in the ontology and there exist multiple paths connecting them to each other.

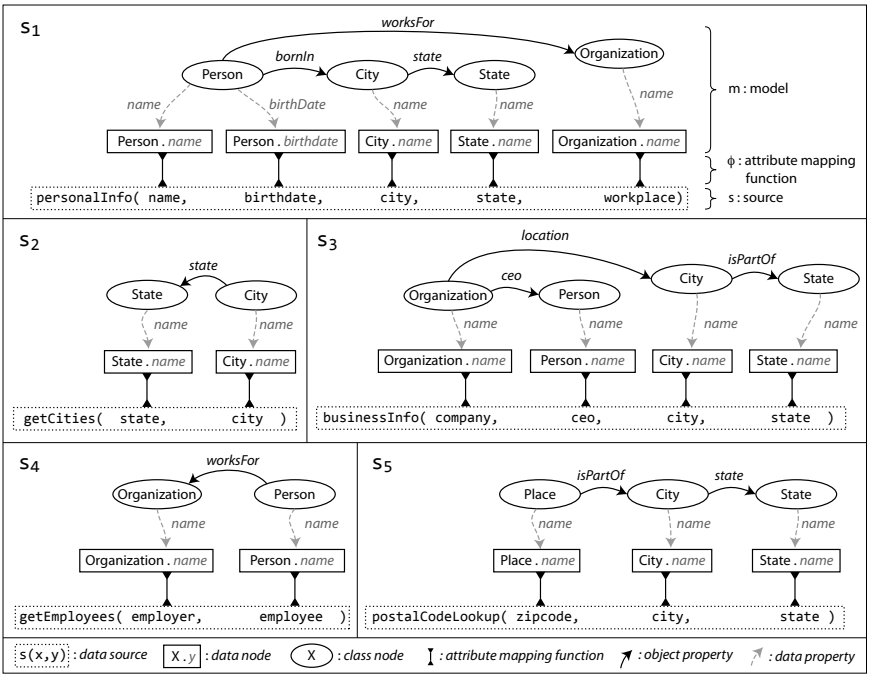


Fig. 2. Source descriptions of sample data sources according to the introduced ontology

In this work, we exploit already modeled sources to build semantic models that are more accurate in terms of the relationships between the source attributes. One of the metrics helping us to build our models is the link popularity, nevertheless, simply using link and node popularity would lead to myopic decisions that select nodes and links that appear frequently in other models without taking into account how these nodes are connected to other nodes in the given models. The main idea of our approach is taking into account the coherency of the patterns and this is much harder to do. Suppose that we have one model containing the link *organizer* between *Event* and *Person* and the link *location* between *Event* and *Place*. We also have several models including *Person* and *Place* (but not *Event*) connected by the relation *bornIn*. If the new source contains *Person*, *Place*, and *Event*, just using the link popularity yields to an incorrect model.

3 Problem Formulation

Having given the example above, we state the problem of learning source descriptions of sources formally.

A *source* is a n-ary relation $s(a_1, \cdots, a_n)$, with a set of attributes $\langle a_1, \cdots, a_n \rangle$, denoted as $Attributes(s)$.

A *semantic model* m is a directed graph containing two types of nodes, *class nodes* and *data nodes*. Class nodes (ovals in Figure 2) correspond to classes in

the ontology and are labeled with class URIs (if v is a class node, $uri(v)$ is URI of the associated class). Data nodes (rectangles in Figure 2), correspond to the range of data properties and are labeled with a pair of URIs: one is the URI of a data property and the the other is the URI of one of the domains of that property e.g., $\langle Person, name\rangle$ or $\langle City, name\rangle$. The links in the graph correspond to ontology properties and are labeled with property URIs (if e is a link, $uri(e)$ represents the URI of the property). In general, a semantic model may contain multiple nodes and links labeled with the same URI.

An *attribute mapping function* ϕ:$Attributes(s)\rightarrow Nodes(m)$ is a mapping from the attributes of source s to a subset of the nodes in m. It can be a partial mapping, i.e, only some of the attributes are connected to the nodes in m.

A *source description* is a triple (s, m, ϕ) where s is a source, m is a semantic model, ϕ is an attribute mapping function that connects the source to the model, and m can be written as a conjunctive query over the predicates of the domain ontology O (in this work, we do not deal with source descriptions involving more complex constructs such as *aggregation*, *union*, or *negation*).

Figure 2 shows the source descriptions for our five example sources. In the figure, ϕ is represented using the inverted arrows symbols (I) connecting attributes in the source to nodes in the model.

The problem of inferring source descriptions can be stated as follows. Let O be a domain ontology and $S = \{(s_1, m_1, \phi_1) \cdots, (s_k, m_k, \phi_k)\}$ a set of source descriptions. Given a new source $\hat{s}$, we want to compose a semantic model $\hat{m}$ and an attribute mapping function $\hat{\phi}$ such that $(\hat{s}, \hat{m}, \hat{\phi})$ is an *appropriate* source description. Clearly, many triples $(\hat{s}, \hat{m}_i, \hat{\phi}_i)$ are well-formed source descriptions, i.e., $\hat{m}_i$ and $\hat{\phi}_i$ are well defined, but only one or a few capture the intended meaning of the data contained in $\hat{s}$. Our goal is to automatically compute $(\hat{s}, \hat{m}, \hat{\phi})$ such that it minimizes the graph edit distance to a source description that the user would deem correct. We evaluate our approach by computing the graph edit distance from $(\hat{s}, \hat{m}, \hat{\phi})$ to a user-defined source description.

4 Learning Semantic Descriptions

The approach we take to learn the source description of a new source aims to characterize a source in terms of the concepts and properties in the domain ontology. In general, the ontology defines a large space (sometimes infinite) of possible source descriptions and without additional information, we do not know which one describes the source more precisely. The main idea here is that data sources in the same domain usually provide overlapping data. Therefore, we can leverage the knowledge of previously described sources to limit the search space and get some hints to hypothesize more plausible candidates.

Our approach has four steps. First, we construct a graph whose main components are the semantic models of the known source descriptions. We use the domain ontology to enrich the graph by adding the nodes and the links that connect these components. Second, we label the source attributes with semantic types. This step is not the focus of this paper and we use an existing technique [12]

to annotate the attributes. Third, we find all possible mappings between the assigned semantic types and nodes of the graph and select the k most promising mappings. We compute a semantic model $\hat{m}$ and an attribute mapping function $\hat{\phi}$ for each mapping to construct the top k source descriptions $(\hat{s}, \hat{m}, \hat{\phi})$. Finally, we define metrics to rank the generated candidates.

4.1 Building a Graph from Known Semantic Models

The central component of our method is a directed weighted graph G built on top of the known semantic models m_i and expanded using the domain ontology O. Similar to the graph of semantic models, G contains both class nodes and data nodes and links corresponding to properties in O. However, the links in G are weighted and they also store a list of the model identifiers, called *support_models*. Algorithm 1 explains how we create G from the known models.

Before we describe the algorithm, we need to define the functions $closure(c)$ and $relations(c_i, c_j)$. For every class c in O, we define $closure(c)$ as the set of classes that either are superclasses of c or can reach c or one of its superclasses by a directed path whose links are object properties. For example, $closure(Person) = \{Organization, Event\}$ because there are the links *ceo* and *organizer* from *Organization* and *Event* to *Person*. As another example, $closure(City) = \{Place, State, Person, Organization, Event\}$. *Place* is in the set because it is a superclass of *City* and the other classes have a path to either *Place* or *City*. We define $relations(c_i, c_j)$ between two class nodes as the properties connecting c_i to c_j. It includes the $subClassOf$ relation and also the properties inherited from the class hierarchy. For instance, $relations(Person, City) = \{bornIn, livesIn\}$ and $relations(City, Place) = \{subClassOf, nearby, isPartOf\}$.

The algorithm has three main parts. In the first part (lines 4-17), we add a component to G for each semantic model m_i that is not a subgraph of the existing components, i.e., m_i introduces a new pattern. If m_i is subsumed by some of the components, we just update the *support_models* of the corresponding links in those components. That means if a pattern appears in k models, all the links of that pattern will have k elements in their *support_models*. Figure 3 illustrates the graph built over $M = \{m_1, m_2, m_3\}$. Although we have three known models, two components are added to G since m_2 is a subgraph of m_1 and we only need to update the *support_models* of the common links between m_1 and m_2.

Next (lines 18-20), we find all the classes that are connected to the current nodes through a path in the ontology. To do this, for every class node v, we calculate $closure(uri(v))$. Then, for each *class_uri* in the resulting set, if G does not already include a node with the same label, we add a new node marked with *class_uri*. In our example in Figure 3, applying this step adds nodes 9 and 11 to the graph because $Event \in closure(City) \cup closure(State) \cup closure(Person)$ and $Place \in closure(City) \cup closure(State)$. In the final part (lines 21-25), we use the ontology properties to connect different components of the graph. We compute $relations(v_i, v_j)$ to find all the possible links between the two class

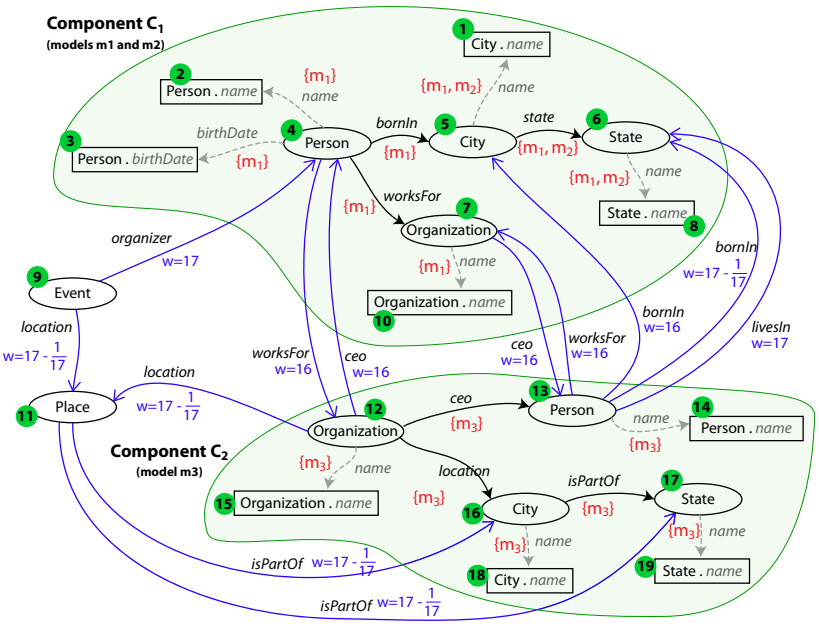


Fig. 3. The graph constructed from $M = \{m_1, m_2, m_3\}$ (semantic models of s_1, s_2, and s_3) and the domain ontology O

nodes v_i and v_j that are not both in the same component. For legibility, we only show some of the links in Figure 3.

Assigning weights to the links of the graph is fundamental in our algorithm. We assign a very low weight $w_{min} = \epsilon$ to all the links inside a component associated with semantic models (black links in Figure 3). For all other links (blue links in Figure 3), we assign a high weight w_{max}. The intuition behind this decision is to produce more coherent models later when we compute the minimal tree. In our example, $w_{max} = 17$ because the total number of links in the set of known models (M) is 17. Regarding the links that do not belong to any component of G (their *support_models* is empty), we use a simple counting mechanism to prefer the more popular ones (lines 42-44). For example, the weight of the link *worksFor* from node 13 to node 7 is 16 because we have seen the same link with the same domain and range in m_1 (the link from node 4 to node 7). As another example, the link *bornIn* from node 13 to node 6 has a weight of $(17 - \frac{1}{17})$. The reason is that although there exist one link with the same label in m_1, the target of the links do not match each other.

4.2 Semantic Labeling of Source Attributes

When presented with a new source $\hat{s}$ whose source description is unknown, the first step is recognizing semantic types of its data. We formally define a *semantic type* to be either an ontology class or a pair consisting of a data property and its

Algorithm 1. *BuildGraph(M, O)*

Input: A set of known semantic models $M = \{m_1, \cdots, m_n\}$ and the domain ontology O
Output: A weighted directed graph G that will be used later in learning semantic descriptions of new sources

$w_{min} \leftarrow \epsilon$
$w_{max} \leftarrow \sum_{i=1}^{n} |Edges(m_i)|$ ▷ w_{max} = total number of the links in M
components $\leftarrow \{\}$ ▷ subgraphs of G corresponding to the semantic models

▷ add the known semantic models to the graph
for each $m_i \in M$ **do**
  if m_i is a subgraph of $c \in$ *components* **then**
    let $c' \subseteq c$ be a subgraph of c that matches m_i
    add "m_i" to the *support_models* of the links in c'
  else ▷ m_i introduces a new pattern
    create a new component c_i by cloning m_i
    for each link $e \in c_i$ **do**
      $support\_models(e) \leftarrow$ "m_i"
      $weight(e) \leftarrow w_{min}$
    end for
    components $\leftarrow$ *components* $\cup\, c_i$
  end if
end for
$G = \bigcup_{c_i \in components} c_i$ ▷ add disjoint components to the graph G

▷ traverse the ontology O to find the classes that do not map to any node in the graph but are connected to them through a path in the ontology
for each *class node* $v \in G$ **do**
  AddClosure(v, G)
end for

▷ use the properties defined in O to join the disconnected components
for $v_i, v_j \in G$ **do**
  if v_i, v_j are both *class nodes* but do not belong to the same pattern **then**
    AddLinks(v_i, v_j, G)
  end if
end for

return G

procedure AddClosure(v, G)
  $ClosureSet \leftarrow closure(uri(v))$
  for each *class_uri* $\in ClosureSet$ **do**
    if there is no node in G labeled with *class_uri* **then**
      add a new node u to G
      $uri(u) \leftarrow$ *class_uri*
    end if
  end for
end procedure

procedure AddLinks(v_i, v_j, G)
  $RelationSet \leftarrow relations(uri(v_i), uri(v_j))$
  for each *property_uri* $\in RelationSet$ **do**
    add a new link e from v_i to v_j
    $uri(e) \leftarrow$ *property_uri*
    $support\_models(e) \leftarrow empty$
    $c_1 \leftarrow \sum_{e' \in E_1} |support\_models(e')|$ where E_1={links of G whose label, source, and target match e}
    $c_2 \leftarrow \sum_{e' \in E_2} |support\_models(e')|$ where E_2={links of G labeled with $uri(e)$}
    $weight(e) \leftarrow min(w_{max} - c_1, w_{max} - c_2/w_{max})$
  end for
end procedure

domain. We use a class as a semantic type for attributes whose values are URIs for instances of a class and for attributes containing automatically-generated database keys that can also be modeled as instances of a class. We use a data property/domain pair as a semantic type for attributes containing literal values. For example, the semantic type for the first attribute of s_4, *employer*, is $\langle Organization, name\rangle$.

We employ a supervised machine learning technique introduced in our previous work [12] to learn semantic types. To achieve high accuracy, we use a Conditional Random Field (CRF) [15] method that uses features extracted both from the attribute names and their values. The CRF is trained with user-specified assignments of attributes to semantic types, specified when the source descriptions for sources s_1 to s_n were constructed.

Applying this method on a new source $\hat{s}$ yields a set of candidate semantic types, each with a confidence value. Our algorithm then selects the top m semantic types for each attribute as an input to the next step of the process. To make the description of the source description construction algorithm simpler, we describe the case of $m = 1$ and then explain how our algorithm can be generalized to handle $m > 1$ (the general case is interesting because it enables the algorithm to cope with situations when the top ranked semantic type is incorrect). Thus, if the new source $\hat{s}$ has n attributes denoted by $Attributes(\hat{s}) = \{a_1, \cdots, a_n\}$, the output of the semantic labeling step is $Labels(\hat{s}) = \{l_1, \cdots, l_n\}$ where l_i is the semantic type of a_i. For example, for s_4 with $Attributes(s_4) = \{employer, employee\}$ we will have $Labels(s_4) = \{\langle Organization, name\rangle, \langle Person, name\rangle\}$.

4.3 Generating Candidate Models

So far, we have annotated the source attributes with semantic types. To build a complete source description we still need to determine the relationships between the attributes. For example, after labeling s_4's attributes, even though $\langle Organization, name\rangle$ and $\langle Person, name\rangle$ are assigned as the semantic types, the relationship between the attributes is not fully determined. It is not clear whether s_4 describes organizations and their employees (using *worksFor* property from *Person* to *Organization*) or organizations and their CEOs (using *ceo* property from *Organization* to *Person*). As we can see, even for simple sources like s_4 having few attributes, the problem of learning an accurate semantic description is a difficult problem. What we propose here is to leverage the knowledge of the known semantic models to discover the most popular and coherent patterns connecting the semantic labels of a new source $\hat{s}$.

The inputs to this step of our algorithm, $Labels(\hat{s})$, are the semantic types assigned to the new source $\hat{s}$ and the graph G, which includes the known semantic models, M, and is expanded using the domain ontology O. The output is a set of candidate source descriptions for the new source $\hat{s}$ where each candidate $(\hat{s}, \hat{m}, \hat{\phi})$ contains the model $\hat{m}$ along with the mapping $\hat{\phi}$ from the source attributes to the nodes of $\hat{m}$. Algorithm 2 shows the steps of our approach.

Algorithm 2. *GenerateCandidates(Labels($\hat{s}$), G)*

Input: A set of semantic types $Labels(\hat{s}) = \{l_1, \cdots, l_n\}$ and the graph G
Output: A set of candidate source descriptions $(\hat{s}, \hat{m}, \hat{\phi})$

1: $Candidates \leftarrow \{\}$

▷ mapping semantic types to the nodes of the graph
2: **for** each $l_i \in Labels(\hat{s})$ **do**
3: $matched(l_i) \leftarrow$ all the nodes in G whose label match l_i
4: **if** $matched(l_i)$ is empty **then** ▷ add new node(s) if no node matches l_i
5: **if** l_i represents a class **then**
6: add a new class node u to G and $uri(u) \leftarrow l_i$
7: $matched(l_i) \leftarrow u$
8: **else** ▷ l_i is in form of $\langle domain, data\ property \rangle$
9: add a new data node v to G and $uri(v) \leftarrow l_i$
10: $matched(l_i) \leftarrow v$
11: add a new class node u to G and $uri(u) \leftarrow domain(l_i)$
12: add a new link e from u to v and $uri(e) \leftarrow data\ property(l_i)$
13: **end if**
14: AddClosure(u, G) ▷ compute the closure of the new node u
15: **for** class nodes v_i, v_j where either v_i or $v_j \in$ new nodes **do**
16: AddLinks(v_i, v_j, G) ▷ connect the new added nodes to the other nodes
17: **end for**
18: **end if**
19: **end for**
20: $MatchedSet \leftarrow \{\{v_1, \cdots, v_n\} | v_i \in matched(l_i)\}$

▷ for each possible mapping from the semantic types to the nodes, we compute the minimal tree that connects those nodes
21: **for** each $\{v_1, \cdots, v_n\} \in MatchedSet$ **do**
22: $SteinerNodes \leftarrow \{v_1, \cdots, v_n\}$
23: $\hat{m} \leftarrow$ SteinerTree$(G, SteinerNodes)$ ▷ find the tree with minimal cost
24: $\hat{\phi} \leftarrow \{\langle a_1, v_1 \rangle, \cdots, \langle a_n, v_n \rangle\}$
25: $Candidates \leftarrow Candidates \cup (\hat{s}, \hat{m}, \hat{\phi})$
26: **end for**

return *Candidates*

In the first part of the algorithm (lines 2-20), we find all the mappings from the semantic types to the nodes of the graph. Since it is possible that a semantic type maps to more than one node in G, more than one mapping might exist from the semantic types to the nodes. For example, if we look into the graph shown in Figure 3, the semantic type $\langle Organization, name \rangle$ maps to nodes 10 and 15 and the semantic type $\langle Person, name \rangle$ maps to nodes 2 and 14. Thus, for $\hat{s} = s_4$, we have four mappings from the semantic types to the graph nodes, $r_1 = \{\langle Organization, name \rangle \rightarrow node\ 10, \langle Person, name \rangle \rightarrow node\ 2\}$, $r_2 = \{\langle Organization, name \rangle \rightarrow node\ 10, \langle Person, name \rangle \rightarrow node\ 14\}$, $r_3 = \{\langle Organization, name \rangle \rightarrow node\ 15, \langle Person, name \rangle \rightarrow node\ 2\}$, and $r_4 = \{\langle Organization, name \rangle \rightarrow node\ 15, \langle Person, name \rangle \rightarrow node\ 14\}$. If a semantic type does not map to any node in the graph, we add a new node to the graph. We use the procedure AddClosure to add the related nodes and then call AddLinks to connect the newly-added nodes to the rest of the nodes in G.

In the next step (lines 21-26), for each set of nodes in each mapping, we find the minimum-cost tree connecting these nodes. Given an edge-weighted graph and a subset of the vertices, called Steiner nodes, the goal is to find the minimum-weight tree that spans all the Steiner nodes. Because the Steiner tree problem

is NP-complete, we use an approximation algorithm [14] with an approximation ratio bounded by $2(1 - 1/l)$, where l is the number of leaves in the optimal Steiner tree. One problem with this algorithm is that if there is large number of mappings from the semantic types to the nodes of the graph, the algorithm becomes inefficient, because we will get a large number of sets as the possible Steiner nodes and we need to run the Steiner tree module over all of them. To solve this problem, we perform an optimization step right after computing the possible mappings. We use a blocking method to eliminate the mappings that are unlikely to generate higher ranked models. This step is not shown in Algorithm 2 to make the algorithm more readable, but we explain it here.

As we will see in the next section, one of the factors to rank the candidate models is their cost. While it is true that the exact cost cannot be calculated until we compute the Steiner tree, the way the links are weighted in G enables us to estimate which sets of Steiner nodes generate lower-cost models. As previously described, all the links inside a known pattern have a very low weight (ϵ). Consequently, sets of Steiner nodes containing larger number of nodes belonging to the same pattern are more likely to yield Steiner trees with lower cost. We apply this idea by sorting all sets of Steiner nodes ($MatchedSet$ in line 20) based on how coherent each set is. For example, considering the four possible mappings we showed earlier for s_4, r_1 and r_4 will be ranked higher than r_2 and r_3, as their matched nodes are part of the same pattern. Once all the node sets are sorted, we pick the top k ones and compute the Steiner tree algorithm only over these sets to generate k candidate models for the new source $\hat{s}$. Figure 4 illustrates the top two candidate models for s_4 and s_5. The inverted arrows ($\downarrow$) depict the mappings from the source attributes to the nodes of the models (ϕ).

The blocking method to reduce the number of mappings also supports generalizing our algorithm to handle the case where each attribute is labeled with a set of possible semantic types rather than only one semantic type (case $m > 1$ discussed in the previous section). To handle this case, we compute the set of all permutations of the semantic types and for each permutation, we find the

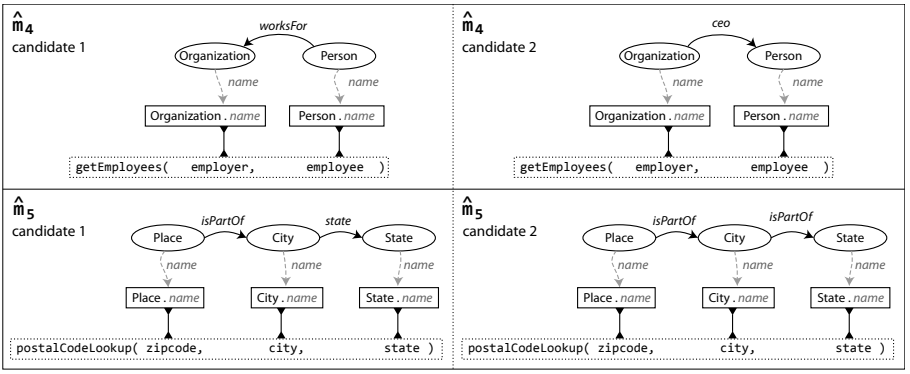


Fig. 4. Top two candidate models generated for s_4 and s_5

possible mappings from the semantic types to the nodes of the graph. Once the mappings are calculated, we apply the blocking technique to get the k most promising node sets. Then, we run the Steiner tree algorithm for each node set to generate k candidate models.

4.4 Ranking Source Descriptions

The final step in learning the semantic description of a source is ranking the candidates produced in the previous step. We define two metrics to rank source descriptions, *coherence* and *cost*. The cost of a candidate $(\hat{s}, \hat{m}, \hat{\phi})$ is the sum of the link weights, $\sum_{e \in \hat{m}} weight(e)$. The goal of defining the coherence is to give more priority to the models containing larger segments from the known patterns. Let $E_p = \{e | e \in \hat{m} \wedge |support\_models(e)| > 0\}$ be the links in model $\hat{m}$ coming from an observed pattern. We partition E_p to create groups of links that belong to the same pattern. More concretely, we define a list $I = (\langle x_1, y_1 \rangle, \cdots, \langle x_n, y_n \rangle)$, where x_i is the size of a group of links sharing a model identifier in their *support_models* (links seen in the same pattern), y_i is the number of model identifiers shared between all of the links in that group, and $\sum_{i=1}^{n} x_i = |E_p|$. In Figure 4, both candidates of $\hat{m}_4$ have $I = \{\langle 3, 1 \rangle\}$, for the first candidate of $\hat{m}_5$ (at the left) $I = \{\langle 3, 2 \rangle\}$, and for the second candidate of $\hat{m}_5$ (at the right), $I = \{\langle 3, 1 \rangle\}$. Note that in $\hat{m}_5$, E_p does not include the links connecting *Place* to the other nodes because *Place* does not exist in any of the observed patterns.

We sort the candidate source descriptions first based on their coherence and then based on their cost. To compare two models regarding the coherence, we compare their coherence lists. We sort the elements of each list descending and then compare the elements one by one. The inequality equation between two elements $z_1 = \langle x_i, y_i \rangle$ and $z_2 = \langle x_j, y_j \rangle$ can be defined as $[z_1 > z_2;\ if\ (x_i > x_j) \vee (x_i = x_j \wedge y_i > y_j)]$. For example, for s_5 the first candidate will be ranked higher than the second one and for s_4, both candidates will be ranked in the same place since they have the same coherence list and the same cost.

5 Evaluation

We evaluated our approach using 17 data sources containing overlapping data (the first column in Table 1 shows the signatures of these sources). We created source descriptions for them manually using the DBpedia, FOAF, GeoNames, and WGS84 ontologies, and used these source descriptions as the gold standard to evaluate our approach. We then used our algorithm to learn a source description for each source given the manually created source descriptions of the other sources as training data (The original source descriptions and the results are available at: `http://isi.edu/integration/data/iswc2013`). Since the semantic labeling is not the focus of this paper, we assume that Algorithm 2 is given the correct semantic type for each attribute (we evaluated the semantic labeling in our previous work [13, 24]).

We used $k = 50$ in the third step of our approach to generate 50 candidate source descriptions and measured the graph edit distance (GED) between the

top ranked description and the manually created one. The results are shown in the second column in Table 1. The value of GED is the (minimum) sum of the costs of the edit operations needed to convert one graph to another. The edit operations include node insertion, node deletion, edge insertion, edge deletion and edge relabeling. Edge relabeling means that we substitute a link between two nodes with another link with the same direction but another URI. We assigned a cost of one to each edit operation.

We compared the results of our approach with the results of Karma [13], a data integration tool that allows users to semi-automatically create source descriptions for sources and services. To make the Karma results comparable to our approach, we also gave Karma the correct semantic types for each attribute. We measured GED between the source description that Karma generates automatically (i.e., without user adjustments) and the gold standard. The results are shown in the third column of Table 1.

The results show that our algorithm generates source descriptions that are more than twice as accurate than Karma-generated ones. Karma learns to assign semantic types, but in this evaluation we gave it the correct semantic types, so Karma was not leveraging any knowledge learned from other source descriptions. Our approach outperformed Karma on all the sources except one. The one incorrect choice is not unexpected since there are cases for which there is no prior evidence or the evidence is misleading. Both systems use a Steiner-tree algorithm to compute their models, so the results show that the learning

Table 1. Evaluation results for learning the semantic descriptions of sample data sources. The second column is the graph edit distance between our hypotheses and the correct source descriptions and the third column is the edit distance between the Karma-generated source descriptions and the correct ones.

Source Signature	**GED** of our models	**GED** of Karma models
nearestCity(lat,lng,city,state,country)	1	6
findRestaurant(zipcode,restaurantName,phone,address)	0	1
zipcodesInCity(city,state,postalCode)	1	3
parseAddress(address,city,state,zipcode,country)	1	6
companyCEO(company,name)	0	1
personalInfo(firstname,lastname,birthdate,brithCity,birthCountry)	1	4
citiesOfState(state,city)	0	1
restaurantChef(restaurant,firstname,lastname)	1	2
capital(country,city)	1	2
businessInfo(company,phone,homepage,city,country,name)	8	10
findSchool(city,state,name,code,homepage,ranking,dean)	6	8
ocean(lat,lng,name)	1	2
employees(organization,firstname,lastname,birthdate)	2	1
education(person,hometown,homecountry,school,city,country)	4	9
postalCodeLookup(zipcode,city,state,country)	1	6
country(lat,lng,code,name)	0	2
administrativeDistrict(city,province,country)	1	4
Total	**29**	**68**

algorithms presented here enable the system to produce more accurate source descriptions.

We also evaluated our approach on five museum datasets modeled using the Europeana EDM, SKOS and FOAF ontologies. The models were created by domain experts for the purpose of creating an integrated dataset. The preliminary results show a 30% improvement, which we believe can be improved further. This improvement is not as large as the improvement on the other test dataset, but it shows that the method works with large, real-world datasets and ontologies.

6 Related Work

The problem of describing semantics of data sources is at the core of data integration [8] and exchange [3]. The main approach to reconcile the semantic heterogeneity among sources consists in defining logical mappings between the source schemas and a common target schema. The R2RML W3C recommendation [6] captures this practice for Semantic Web applications. Although these mappings are declarative, defining them requires significant technical expertise, so there has been much interest in techniques that facilitate their generation.

The mapping generation problem is usually decomposed in a *schema matching* phase followed by *schema mapping* phase [4]. Schema matching [20] finds correspondences between elements of the source and target schemas. For example, iMAP [7] discovers complex correspondences by using a set of special-purpose searchers, ranging from data overlap, to machine learning and equation discovery techniques. We use our previous work on semantic labeling [12], which considers attributes that map to the same semantic type as potential matches. Schema mapping defines an appropriate transformation that populates the target schema with data from the sources. Mappings may be arbitrarily procedures, but of greater interest are declarative mappings expressible as queries in SQL, XQuery, or Datalog. These mapping formulas are generated by taking into account the schema matches and schema constraints. There has been much research in schema mapping, from the seminal work on Clio [10], which provided a practical system and furthered the theoretical foundations of data exchange [11] to more recent systems that support additional schema constraints [17]. Alexe et al. [1] generate schema mappings from examples of source data tuples and the corresponding tuples over the target schema. Karma [13] and An et al. [2] generate mappings into ontologies, suggested by exploring low-cost Steiner trees that connect matching semantic types within a graph derived from the target ontology. Karma allows the user to correct the mappings interactively.

Our work in this paper is complementary to these schema mapping techniques. Instead of focusing on satisfying schema constraints, we analyze known source descriptions to propose mappings that capture more closely the semantics of the target source, in ways that schema constraints could not disambiguate. For example, suggesting that a *worksFor* relationship is more likely than *ceo* in a given domain. Moreover, following Karma, our algorithm can incrementally refine the mappings based on user feedback and improve future predictions. Carman and

Knoblock [5] also use known source descriptions to generate a LAV mapping for an unknown target source. However, a limitation of that work is that their approach could only learn descriptions that were conjunctive combinations of known source descriptions. By exploring paths in the domain ontology, in addition to patterns in the known sources, we can hypothesize target mappings that are more general than previous source descriptions or their combinations.

Semantic annotation of services [21, 22] and more recently of web tables [16, 18, 25] has also received attention. Most of this work learns types for services parameters or table columns, but is limited in learning relationships. Limaye et al [16] generate binary relationships leveraging the Yago ontology.

Ontology alignment [9] usually considers alignments between individual classes, so it is more applicable to the matching phase. However, Parundekar et al. [19] use an extensional approach to discover alignments between conjunctions and disjunctions of classes from linked data ontologies.

7 Discussion

We presented a novel approach to automatically learn the semantic description of a new source given a set of known semantic descriptions as the training set and the domain ontology as the background knowledge. The learned semantic descriptions explicitly represent the relationships between the source attributes in addition to their semantic types. These precise descriptions of data sources makes it possible to automatically integrate the data across sources and provides rich support for source discovery.

In our approach we build a graph whose main components are the known semantic descriptions expanded using the domain ontology. Next, we use a machine learning technique to label the attributes of the new source with classes and properties of the ontology. We find the possible one-to-one mappings from the semantic types to the nodes of the graph and calculate the top k promising mappings. Then, we build a tree over each mapping to generate k candidate models. Finally, we score the candidates to output a ranked list of the most plausible semantic models. The evaluation results showed that our algorithm generates models that are more accurate than Karma, a state of the art tool to semi-automatically model data sources.

The graph construction in the presented algorithm is an incremental process, i.e., we augment the graph with a new component when a new known model is presented to the system. The algorithm that we use to compute the Steiner tree is an approximation algorithm whose complexity is $O(|S||V|^2)$ where V is the set of nodes in the graph and S is a subset of the nodes (size of S is equal to the number of the new source attributes). Thus, computing the candidate models might be challenging when the size of the graph is very large in terms of the number of the nodes, even though we are using a polynomial time algorithm. In future work, we plan to investigate the idea of creating a more compact graph by consolidating the overlapping segments of the known semantic models. This will reduce the number of nodes added to the graph when a new pattern is given to the system. We also plan to integrate our approach into Karma in order to

suggest more accurate semantic models to users. This will make it possible to automatically produce source descriptions with minimal user input.

Acknowledgements. This research is based upon work supported in part by the National Science Foundation under award number IIS-1117913 and in part by the NIH through the following NIGMS grant: U24 GM104203 Bio-Informatics Research Network Coordinating Center (BIRN-CC). The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of NSF or any person connected with them.

References

1. Alexe, B., ten Cate, B., Kolaitis, P.G., Tan, W.C.: Designing and Refining Schema Mappings via Data Examples. In: SIGMOD, Athens, Greece, pp. 133–144 (2011)
2. An, Y., Borgida, A., Miller, R.J., Mylopoulos, J.: A Semantic Approach to Discovering Schema Mapping Expressions. In: Proceedings of the 23rd International Conference on Data Engineering (ICDE), Istanbul, Turkey, pp. 206–215 (2007)
3. Arenas, M., Barcelo, P., Libkin, L., Murlak, F.: Relational and XML Data Exchange. Morgan & Claypool, San Rafael (2010)
4. Bellahsene, Z., Bonifati, A., Rahm, E.: Schema Matching and Mapping, 1st edn. Springer (2011)
5. Carman, M.J., Knoblock, C.A.: Learning Semantic Definitions of Online Information Sources. Journal of Artificial Intelligence Research 30(1), 1–50 (2007)
6. Das, S., Sundara, S., Cyganiak, R.: R2RML: RDB to RDF Mapping Language. W3C Recommendation (September 27, 2012), `http://www.w3.org/TR/r2rml/`
7. Dhamankar, R., Lee, Y., Doan, A., Halevy, A., Domingos, P.: iMAP: Discovering Complex Semantic Matches between Database Schemas. In: International Conference on Management of Data (SIGMOD), New York, NY, pp. 383–394 (2004)
8. Doan, A., Halevy, A., Ives, Z.: Principles of Data Integration. Morgan Kauffman (2012)
9. Euzenat, J., Shvaiko, P.: Ontology Matching. Springer, Heidelberg (2007)
10. Fagin, R., Haas, L.M., Hernández, M., Miller, R.J., Popa, L., Velegrakis, Y.: Clio: Schema Mapping Creation and Data Exchange. In: Borgida, A.T., Chaudhri, V.K., Giorgini, P., Yu, E.S. (eds.) Mylopoulos Festschrift. LNCS, vol. 5600, pp. 198–236. Springer, Heidelberg (2009)
11. Fagin, R., Kolaitis, P.G., Miller, R.J., Popa, L.: Data Exchange: Semantics and Query Answering. Theoretical Computer Science 336(1), 89–124 (2005)
12. Goel, A., Knoblock, C.A., Lerman, K.: Exploiting Structure within Data for Accurate Labeling Using Conditional Random Fields. In: Proc. ICAI (2012)
13. Knoblock, C.A., et al.: Semi-Automatically Mapping Structured Sources into the Semantic Web. In: Simperl, E., Cimiano, P., Polleres, A., Corcho, O., Presutti, V. (eds.) ESWC 2012. LNCS, vol. 7295, pp. 375–390. Springer, Heidelberg (2012)
14. Kou, L., Markowsky, G., Berman, L.: A Fast Algorithm for Steiner Trees. Acta Informatica 15, 141–145 (1981)
15. Lafferty, J., McCallum, A., Pereira, F.: Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In: Proceedings of the 18th International Conference on Machine Learning (2001)

16. Limaye, G., Sarawagi, S., Chakrabarti, S.: Annotating and Searching Web Tables Using Entities, Types and Relationships. PVLDB 3(1), 1338–1347 (2010)
17. Marnette, B., Mecca, G., Papotti, P., Raunich, S., Santoro, D.: ++Spicy: an OpenSource Tool for Second-Generation Schema Mapping and Data Exchange. In: Procs. VLDB, Seattle, WA, pp. 1438–1441 (2011)
18. Mulwad, V., Finin, T., Joshi, A.: A Domain Independent Framework for Extracting Linked Semantic Data from Tables. In: Ceri, S., Brambilla, M. (eds.) Search Computing III. LNCS, vol. 7538, pp. 16–33. Springer, Heidelberg (2012)
19. Parundekar, R., Knoblock, C.A., Ambite, J.L.: Discovering Concept Coverings in Ontologies of Linked Data Sources. In: Cudré-Mauroux, P., et al. (eds.) ISWC 2012, Part I. LNCS, vol. 7649, pp. 427–443. Springer, Heidelberg (2012)
20. Rahm, E., Bernstein, P.: A Survey of Approaches to Automatic Schema Matching. VLDB Journal 10(4) (2001)
21. Saquicela, V., Vilches-Blazquez, L.M., Corcho, O.: Lightweight Semantic Annotation of Geospatial RESTful Services. In: Antoniou, G., Grobelnik, M., Simperl, E., Parsia, B., Plexousakis, D., De Leenheer, P., Pan, J. (eds.) ESWC 2011, Part II. LNCS, vol. 6644, pp. 330–344. Springer, Heidelberg (2011)
22. Sheth, A.P., Gomadam, K., Ranabahu, A.: Semantics Enhanced Services: METEOR-S, SAWSDL and SA-REST. IEEE Data Eng. Bulletin 31(3), 8–12 (2008)
23. Szekely, P., Knoblock, C.A., Yang, F., Zhu, X., Fink, E.E., Allen, R., Goodlander, G.: Connecting the Smithsonian American Art Museum to the Linked Data Cloud. In: Cimiano, P., Corcho, O., Presutti, V., Hollink, L., Rudolph, S. (eds.) ESWC 2013. LNCS, vol. 7882, pp. 593–607. Springer, Heidelberg (2013)
24. Taheriyan, M., Knoblock, C.A., Szekely, P., Ambite, J.L.: Rapidly Integrating Services into the Linked Data Cloud. In: Cudré-Mauroux, P., et al. (eds.) ISWC 2012, Part I. LNCS, vol. 7649, pp. 559–574. Springer, Heidelberg (2012)
25. Venetis, P., Halevy, A., Madhavan, J., Paşca, M., Shen, W., Wu, F., Miao, G., Wu, C.: Recovering Semantics of Tables on the Web. Proc. VLDB Endow. 4(9), 528–538 (2011)

QODI: Query as Context in Automatic Data Integration

Aibo Tian, Juan F. Sequeda, and Daniel P. Miranker

Department of Computer Science, The University of Texas at Austin
Austin, Texas, USA
{atian,jsequeda,miranker}@cs.utexas.edu

Abstract. QODI is an automatic ontology-based data integration system (OBDI). QODI is distinguished in that the ontology mapping algorithm dynamically determines a partial mapping specific to the reformulation of each query. The query provides application context not available in the ontologies alone; thereby the system is able to disambiguate mappings for different queries. The mapping algorithm decomposes the query into a set of paths, and compares the set of paths with a similar decomposition of a source ontology.

Using test sets from three real world applications, QODI achieves favorable results compared with AgreementMaker, a leading ontology matcher, and an ontology-based implementation of the mapping methods detailed for Clio, the state-of-the-art relational data integration and data exchange system.

Keywords: QODI, Ontology-based data integration, Query-specific ontology mapping.

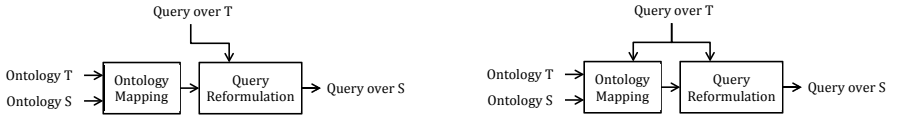
1 Introduction

Web-wide integration of structured data is being enabled by the emerging Semantic Web protocols that specify uniform query interfaces to the databases included in the deep web [10]. These developments were recently boosted by W3C ratification of standards for publishing relational database content as RDF[1]. The scope of the deep web underscores the need for automating data integration. The Semantic Web technology stack enables an ontology to serve as a federating data model. Heterogeneous distributed database systems that use an ontology as a federating data model are called ontology-based data integration systems (OBDI).

This paper details the mapping algorithms of Query-driven Ontology-based Data Integration (QODI). QODI is currently deployed as the mediator of a faceted search system over RNA databases[2]. QODI considers two OWL ontologies: the target ontology, which is the federating data model, and the source ontology. SPARQL queries are issued over the target ontology by users, and translated to the queries over the source ontology. Although QODI is designed to integrate RDF data, a primary motivation is the integration of relational data. Several of our test cases comprise relational databases virtualized as RDF, and SQL schemas translated to ontologies [13,14].

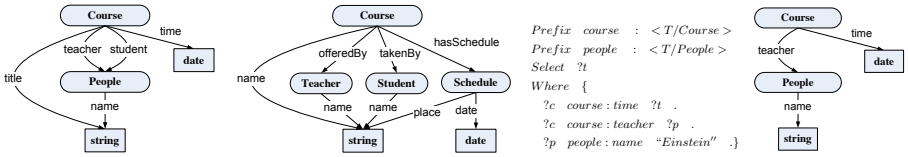
[1] http://www.w3.org/2001/sw/wiki/RDB2RDF
[2] http://ribs.csres.utexas.edu/ontoexplorer

H. Alani et al. (Eds.): ISWC 2013, Part I, LNCS 8218, pp. 624–639, 2013.



(a) Traditional (b) The proposed, QODI

Fig. 1. Diagram of OBDI systems with traditional and the proposed ontology mapping



(a) Ontology T (b) Ontology S (c) SPARQL query (d) Query graph

Fig. 2. Example ontologies and SPARQL query about the domain of course. Oval vertices represent classes, and rectangular vertices are datatypes. Edges represent object properties, or datatype properties. The SPARQL query asks for the time of any course taught by "Einstein".

In the typical organization of an OBDI system, ontology mapping is a separate and prerequisite step of query reformulation (see Figure 1(a)). Ontology matchers may be introduced to automatically determine corresponding entities [3,6]. In this paper, an entity refers to a class or a property. We tested AgreementMaker [5], one of the top finishers in 2010 Ontology Alignment Evaluation Initiative (OAEI) [1]. The highest accuracy of AgreementMaker on our test sets is less than 42%. Inspection of these results revealed two dominant challenges: *ambiguous mapping* and *missing mapping*. We create a small example to illustrate the challenges. Figure 2 shows a target ontology T, a source ontology S, and a SPARQL query q which asks for the time of any course that is taught by Einstein.

The Ambiguous Mapping Challenge: An entity in the target ontology has an ambiguous mapping if it can be mapped to more than one entity in the source ontology, and the correct choice is dependent on the application. In other words, there is not enough information in the ontologies alone to determine a correct mapping. An example of ambiguous mapping considers that *name* of class *People* in T can be mapped to *name* of either class *Teacher* or *Student* in S. There is no basis for preferring one mapping or another. However, considering query q, clearly *Teacher* is preferred.

Some matchers would identify this example as a *complex mapping* such that *name* of *People* maps to the union of both *name* of *Teacher* and *Student*, since both *Teacher* and *Student* can be identified as subclasses of *People*. In isolation of an application, the logic of the complex mapping is correct. But, if the example query is reformulated using both alternatives, the translated query will return the time of any course that either taught or taken by Einstein. The reformulation is incorrect. Thus, only after the query is known, is it possible to disambiguate the mapping.

Ambiguous mappings occur often. In our real world test sets, two out of three domains have ambiguity. In those, 10% to 30% of the query workload displays ambiguity.

The Missing Mapping Challenge: Some entities do not have any mapping, such as the class *Schedule* and property *hasSchedule* in S. Matchers can find out that both *Course* in T and S are mapped, and *time* and *date* are mapped. However, *Schedule* and *hasSchedule*, which are in the middle of the path from *Course* to *date*, do not have any mapping. Query q cannot be reformulated for execution on S without including *Schedule* and *hasSchedule*.

We formally define *query-specific ontology mapping*. For each input query, the system determines a partial ontology mapping sufficient to reformulate the specific query. In effect, a query becomes a third argument to the ontology mapping algorithm (see Figure 1(b)). Note that using the query as context requires no extra input from users or experts. In QODI, both the input query and the source ontology are decomposed into paths, and mapping concerns identifying correspondences between paths instead of entities. Path similarity is estimated based on the feature vectors that are generated by representing each path as a bag of entity labels. Given an input query, QODI searches for a subgraph of the source ontology, such that the set of path correspondences has the highest confidence. QODI exploits efficient heuristic search algorithms, which guarantee to find an optimal solution. By leveraging queries to provide context, the ambiguous mapping challenge is resolved. Since the path similarity is not dependent on the precise alignment of entities, the missing mapping challenge is resolved.

In our running example, the path that contains *People* and *name* in query q also contains *teacher*. In ontology S, the path with *Teacher* has higher string vector similarity than the one with *Student*. The two path correspondences for the query should be:

$$\{\text{Course,teacher,People,name,string}\} = \{\text{Course,offeredBy,Teacher,name,string}\}$$
$$\{\text{Course,time,date}\} = \{\text{Course,hasSchedule,Schedule,date,date}\}$$

QODI is evaluated on three real world application domains: Life Science, Bibliography, and Conference Organization. QODI outperforms all baselines on all test cases.

2 Problem Definition

The following section begins with graph definitions and culminates with the formal definition of the mapping problem.

2.1 Basic Graph Definition

An *ontology graph* is a representation of an ontology as a directed labeled graph, where classes and datatypes are vertices, and properties are edges (see Figure 2). Target and source ontologies are distinguished as T and S, respectively. These notations are used interchangeably to denote ontologies and ontology graphs. To simplify handling inheritance relationships, rather than coding the logic of inheritance into the path-related algorithms, an ontology graph is expanded by replicating properties. If the domains or ranges of a property have subclasses, new edges with the same label as that property are created for each subclass.

Definition 1 (source and sink). *In a directed labeled graph G, a source is a vertex with 0 in-degree, and a sink is a vertex with 0 out-degree. The sets of all sources and sinks of G are denoted $SOURCE_G$ and $SINK_G$, respectively.*

Definition 2 (ss-path). *A source-to-sink path or ss-path is a path from a vertex v_1 to a vertex v_2 in a directed labeled graph G, where v_1 is a source and v_2 is a sink of G.*

For convenience, we represent a path p as an ordered list of vertices and edges, and define the length, denoted as $|p|$, as the sum of the number of vertices and edges in p.

Definition 3 (ss-path-set). *The set of all possible ss-paths from source v_1 to sink v_2 in a directed labeled graph G is called an ss-path-set (denoted as SS-PATH-SET$_{G,v_1,v_2}$).*

Definition 4 (graph-ss-path-set). *Given a directed labeled graph G, the set of all ss-paths (denoted as GRAPH-SS-PATH-SET$_G$) is the union of all ss-path-sets from all sources to all sinks in G.*

Definition 5 (query graph). *Given a SPARQL query q over ontology T, a query graph (denoted as Q) is a subgraph of T that corresponds to q.*

The query graph of the SPARQL query in Figure 2(c) is shown in Figure 2(d).

2.2 Assumptions

Basic assumptions are as follows:

1. All object properties and datatype properties have domains and ranges. This assumption simplifies the construction of ontology graphs. High quality manually designed ontologies will detail domains and ranges. Ontologies automatically translated from relational schemas include domains and ranges [13,14].

2. We consider conjunctive SPARQL queries in the SELECT query form, and exclude variables from the predicates of triple patterns. For each variable, the class, which is the type that the variable is binding to, either can be inferred from the domains or ranges of predicates or is provided by rdf:type. Given these assumptions, there exists only one query graph for each query. If multiple query graphs are allowed, each of them can be mapped separately. For simplicity, we leave the relaxing of these assumptions for future work.

3. The sinks of a query graph only represent datatypes. This paper concerns ontologies that describe database content and queries that retrieve information from databases. Retrieving database data ultimately requires the rewriting of datatype properties.

2.3 Query-Specific Ontology Mapping

The following definitions define query-specific ontology mapping, which is the core problem of this paper. An ss-path correspondence records the mapping confidence between two ss-paths.

Definition 6 (ss-path correspondence). *Given two directed labeled graphs G and G', an ss-path correspondence between two ss-paths p and p' (denoted by $\pi_{p,p'}$) is $< p, p', \alpha_{\pi_{p,p'}} >$, such that $p \in$ GRAPH-SS-PATH-SET$_G$, $p' \in$ GRAPH-SS-PATH-SET$_{G'}$, and $\alpha_{\pi_{p,p'}}$ is a confidence measure between 0 and 1.*

A *match candidate* is a set of ss-path correspondences between the ss-paths in the query graph, and the ss-paths in a subgraph of the source ontology graph.

Definition 7 (match candidate). *Given a query graph Q, a match candidate $\Omega_{Q,G}$ is a set of ss-path correspondences between the ss-paths in Q and the ss-paths in a graph G, which is a subgraph of the source ontology S, if the following conditions are satisfied:*

- *The sinks of G are datatypes;*
- *for each ss-path $p \in$ GRAPH-SS-PATH-SET$_Q$, there exists exactly one ss-path correspondence $\pi_{p,p'} \in \Omega_{Q,G}$, where $p' \in$ GRAPH-SS-PATH-SET$_G$;*
- *for each ss-path $p' \in$ GRAPH-SS-PATH-SET$_G$, there exists ss-path correspondences $\pi_{p,p'} \in \Omega_{Q,G}$, where $p \in$ GRAPH-SS-PATH-SET$_Q$;*
- *for each pair of ss-paths $p_1, p_2 \in$ GRAPH-SS-PATH-SET$_Q$, if they share a common source, then the two corresponding ss-paths $p'_1, p'_2 \in$ GRAPH-SS-PATH-SET$_G$ also share a common source, where $\pi_{p_1,p'_1} \in \Omega_{Q,G}$, $\pi_{p_2,p'_2} \in \Omega_{Q,G}$.*

Definition 7 contains several constraints. First, all sinks of G are required to be datatypes, because the sinks of the query graph Q are also datatypes. Second, we are interested in a one-to-one mapping, which restricts each ss-path in Q to be contained in exactly one correspondence. Third, if the ss-paths in Q share a source, the mapped ss-paths in G also share a source. We assign a confidence measure $\beta_{\Omega_{Q,G}}$, which is defined as the product of all ss-path correspondence confidence measures:

$$\beta_{\Omega_{Q,G}} = \prod_{\pi_{p,p'} \in \Omega_{Q,G}} \alpha_{\pi_{p,p'}}$$

The task of query-specific ontology mapping, *q-mapping*, is to find the match candidate with the highest confidence.

Definition 8 (q-mapping). *Given two ontology graphs T, S, and a SPARQL query q over T, the query-specific ontology mapping (denoted as q-mapping(T,S,q)) is the set of ss-path correspondences $\Omega_{Q,\bar{G}}$, where Q is the query graph, and $\bar{G}$ is a subgraph of S, such that $\Omega_{Q,\bar{G}}$ is a match candidate, and $\beta_{\Omega_{Q,\bar{G}}} = \max_{G \subseteq S} \beta_{\Omega_{Q,G}}$.*

3 QODI: Mapping and Reformulation

The goals of mapping include defining a similarity score between two ss-paths, and determining the highest scoring ss-path correspondences without an exhaustive search.

3.1 ss-path Similarity Measure

The ss-path similarity measure must be able to disambiguate uncertain mappings. Given a pair of ss-paths, the similarity is defined as a product of four factors: similarity between source classes, similarity between datatype properties, similarity between path labels, and a penalty for path length differences. The source class and datatype property determine the two ends of an ss-path. A path label, containing the labels of all entities except datatypes in an ss-path, is used to disambiguate uncertain mappings.

Similarity estimation of source classes and datatype properties has been well studied in prior work [5,12,6]. Any existing method may be used for this component. In the experiments, we evaluated both simple string distance and sophisticated ontology matchers. The similarity between all classes and datatype properties can be computed beforehand and stored as similarity matrices for lookup.

We borrow techniques from information retrieval to measure the similarity between path labels. For an ss-path, we process the labels of all entities except datatypes in the path using linguistic processing, and add the processed strings to a list. The linguistic processing includes tokenization by punctuation, numbers, and uppercase letters (if the letter is not preceded by an uppercase letter); stop words removal; and stemming (using SimPack[3]). All strings are converted to lowercase. A feature vector is generated by indexing the list of strings, and using frequencies as features. Given that different labels may contain a different number of tokens, the frequency of a token is set to one over the number of tokens in a label. The path label similarity, S_L, is computed as the intersection between the two feature vectors.

$$S_L(p,p') = \frac{\sum_{i=1}^{m} \min(\mathbf{f}_i(p), \mathbf{f}_i(p'))}{\sum_{i=1}^{m} \mathbf{f}_i(p) + \mathbf{f}_i(p') - \min(\mathbf{f}_i(p), \mathbf{f}_i(p'))} \tag{1}$$

where $\mathbf{f}_i(p)$ is the ith element of the feature vector of ss-path p, and m is the dimension of the feature vectors.

If two paths are similar, their lengths may not have a large difference. We use an exponential function to penalize the path length difference. The ss-path similarity measure, S_{SS}, is defined as,

$$S_{SS}(p,p') \;=\; S_C(p,p')^{\frac{1}{n_p}} \cdot S_D(p,p') \cdot S_L(p,p') \cdot e^{-\eta \cdot |\,|p|-|p'|\,|} \tag{2}$$

where S_C and S_D are similarity measures for source classes and datatype properties, which are provided by matchers. $|p|$ is the length of path p, and η is a non-negative real number. n_p is the number of ss-paths in the query graph that share the same source with p. n_p is introduced because the same similarity between sources will be multiplied n_p times when measuring the confidence of a match candidate.

3.2 q-mapping

We denote the set of all possible match candidates of query graph Q as $\mathcal{M}_Q$. $\bar{G}$, which is the subgraph of S involved in the match candidate with the highest similarity, is determined by maximizing the confidence measure. q-mapping(T,S,q) is the set of ss-path correspondences $\Omega_{Q,\bar{G}}$ between Q and $\bar{G}$.

$$\begin{aligned}\bar{G} &= \underset{\Omega_{Q,G} \in \mathcal{M}_Q}{\arg\max}\ \beta_{\Omega_{Q,G}} \\ &= \arg_{G \subseteq S} \{ \prod_{c \in SOURCE_Q} \{ \max_{c' \in SOURCE_G} \{ \\ &\qquad \prod_{p \in \text{SS-PATH-SET}_{Q,c,*}} \{ \max_{p' \in \text{SS-PATH-SET}_{G,c',*}} S_{SS}(p,p') \} \} \} \}\end{aligned} \tag{3}$$

[3] files.ifi.uzh.ch/ddis/oldweb/ddis/research/simpack/

where $\beta_{\Omega_{Q,G}}$ is the confidence measure of the match candidate, and SS-PATH-SET$_{G,c,*}$ represents the set of all ss-paths with source c in G.

Equation (3) specifies the mapping as: for each source vertex in the query graph, find a vertex in the source ontology as a source vertex, such that the product of all ss-path similarities is the maximum.

3.3 Solving the Maximization

Equation (3) does not specify how to solve the maximization. A naive algorithm may score all possible match candidates. However, the number of all possible paths can be exponential in the number of vertices for acyclic ontology graphs, and is infinite for cyclic ontology graphs. It is infeasible to compute similarity between all pairs of paths. Thus, we employ heuristic search algorithms to reduce the computation.

We decompose the search problem into two phases: 1) given an ss-path in the query graph and a vertex in S, search for the ss-path in S with the given vertex as source that has the highest similarity; 2) given a set of ss-paths that share a source in the query graph, find a set of paths in S that share a source and have the highest product of similarities. Phase 1) is a subproblem of 2). Thus, we solve 1) then 2).

Phase 1) can be solved by a heuristic search algorithm similar to A* search. A* is commonly applied to find a minimal cost path in a graph [9]. A* requires a function that computes the cost of a partial path, and a heuristic cost function that estimates the cost of completing a path. The search is guaranteed to terminate with an optimal path if the heuristic is admissible. We cannot exploit the traditional structure of A* search. Our definition of path similarity considers all labels in a path as a bag of words. Thus, we can not decompose a partially computed answer into the sum of two functions. We define a single function that, given a partial path, will never overestimate the cost of a complete optimal path. With similar proof as A* search, our heuristic search is guaranteed to find an optimal path. The implementation of the search algorithm remains largely unchanged. Search states, representing partial paths, are saved in an open-list $\mathcal{P}$. $\mathcal{P}$ is initialized by the path that only contains one vertex (the given vertex). The paths in $\mathcal{P}$ are sorted in ascending order using our heuristic function. The search terminates when a path $\bar{p}$ containing a sink (datatype) is pulled from $\mathcal{P}$.

We introduce two techniques to help create the heuristic cost function. First, the similarity between datatype properties (S_D), which is a factor of S_{SS}, is considered at the beginning of the search. A datatype property is the last edge in an ss-path, and connects to a datatype. Thus, a large amount of computation can be potentially wasted by the search before discovering the similarity between datatype properties is low. To address this, $\mathcal{P}$ is initialized by a set of paths, each of which only contains the given vertex and only leads to the sink through a specific datatype property. Following that, S_D is a constant for each path. S_C is also a constant, since the source vertex is given. Only the cost of adding new vertices and edges to the path needs to be considered.

The second technique is a preprocessing step that associates reachable label sets and shortest path lengths to each class. We define a reachable label set from a vertex through a datatype property as the union of the path labels of all possible paths from the vertex to a datatype through the datatype property. Each vertex of S is associated with the reachable label sets, from itself through each datatype property. Figure 3

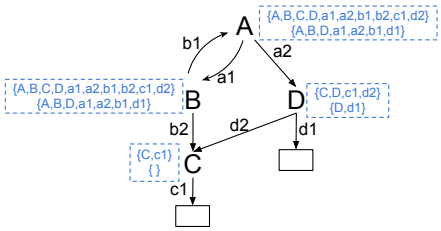


Fig. 3. An example ontology graph with reachable label sets. The dashed boxes around each non-sink vertex contain the reachable label sets through the two datatype properties $c1$ and $d1$. For example, the reachable label set from C through $c1$ is $\{C, c1\}$, and through $d1$ is empty.

illustrates an example of reachable label sets. The reachable label sets are computed by recursively propagating the reachable label sets of each vertex to its parents. The algorithm terminates when the reachable label sets are not changed for all vertices. The worst case complexity of this algorithm is quadratic in the number of vertices. Given that a reachable label set is a superset of the labels that may appear in an optimal path, a heuristic can be defined to guarantee the optimality. In addition, each vertex of S is also associated with the lengths of the shortest paths from itself to datatypes through each datatype property. The lengths of shortest paths are also used in the heuristic. Note that the preprocessing only need run once.

We denote the ss-path in the query graph as r, the path in S that needs heuristic scoring as p, the last element of p as x, and the objective datatype as e. The reachable label set from x to e is denoted as $L_{x,e}$, and the length of the shortest path from x to e is denoted as $l_{x,e}$. The heuristic cost function h is defined as follows:

$$h(p) = -S_C(r,p)^{\frac{1}{n_r}} \cdot S_D(r,p) \cdot \frac{\sum_{i=1}^{m} \min(\mathbf{f}_i(r), \mathbf{f}_i(p) + \bar{\mathbf{f}}_i(r,p,L_{x,e}))}{\sum_{i=1}^{m} \mathbf{f}_i(r) + \mathbf{f}_i(p) - \min(\mathbf{f}_i(r), \mathbf{f}_i(p))} \cdot e^{-\eta \cdot g(r,p,l_{x,e})} \tag{4}$$

where n_r is the number of paths in the query graph that share the same source as r, and $\mathbf{f}_i(p)$ is the ith element of the feature vector of path p. $\bar{\mathbf{f}}_i(r,p,L_{x,e})$ and $g(r,p,l_{x,e})$ are:

$$\bar{\mathbf{f}}_i(r,p,L_{x,e}) = \begin{cases} \max(\mathbf{f}_i(r) - \mathbf{f}_i(p),\ 0) & \text{, if } x \neq e \text{ and string } i \in L_{x,e} \\ 0 & \text{, otherwise} \end{cases} \tag{5}$$

$$g(r,p,l_{x,e}) = \begin{cases} \max(|p| + l_{x,e} - 1 - |r|,\ 0) & \text{, if } x \neq e \\ |\ |p| - |r|\ | & \text{, if } x = e \end{cases} \tag{6}$$

Comparing (4) with (2), h is derived from the negation of S_{SS} by substituting a real path by an estimation. Let us denote the path as $\bar{p}$, when the search terminates. Based on the termination condition, $x = e$. Substituting x with e, $h(\bar{p}) = -S_{SS}(r,\bar{p})$. The following lemma and theorem prove that $\bar{p}$ is the best scoring path. Lemma 1 corresponds to the proof of admissibility of the heuristic in A* search.

Lemma 1. *Suppose the search has not terminated. For any optimal path $\tilde{p}$, there exists a path p in the priority queue $\mathcal{P}$, which can be expanded to $\tilde{p}$, such that $h(p) \leq h(\tilde{p})$.*

Proof. h is the negation of a product of four non-negative factors. We will prove that each factor of $h(p)$ is greater than or equal to the corresponding factor of $h(\tilde{p})$. Then $h(p) \leq h(\tilde{p})$.

The first two factors, S_C and S_D, are the same for both p and $\tilde{p}$.

Consider the third factor. Denote the last element in path p as x. The reachable label set, $L_{x,e}$, contains the labels of all possible paths from x to e, including those in $\tilde{p}$. Per the definition of $\overline{\mathbf{f}}_i$, the numerator in $h(p)$ is greater than or equal to that in $h(\tilde{p})$. Since p is a sub-path of $\tilde{p}$, the denominator in $h(p)$ is less than or equal to that in $h(\tilde{p})$. Thus the third factor of $h(p)$ is greater than or equal to the third factor of $h(\tilde{p})$.

Consider the fourth factor. $l_{x,e}$ is the length of the shortest path from x to e, so $|p| + l_{x,e} - 1 \leq |\tilde{p}|$. Consider two cases:

1. $|p|+l_{x,e}-1 \geq |r|$. Then $g(r, p, l_{x,e}) = |p|+l_{x,e}-1-|r|$, and $g(r, \tilde{p}, l_{e,e}) = |\tilde{p}|-|r|$. Thus, $g(r, p, l_{x,e}) \leq g(r, \tilde{p}, l_{e,e})$.
2. $|p|+l_{x,e}-1 < |r|$. Then $g(r, p, l_{x,e}) = 0$, and $g(r, \tilde{p}, l_{e,e}) \geq 0$. Thus, $g(r, p, l_{x,e}) \leq g(r, \tilde{p}, l_{e,e})$.

Thus the fourth factor of $h(p)$ is greater than or equal to the fourth factor of $h(\tilde{p})$.

Theorem 1. *When the search terminates, the path $\bar{p}$ is an optimal path.*

The proof of Theorem 1 can be derived from the proof of the similar theorem for A* search by substituting the sum of the two cost functions with our heuristic $h(p)$ [9].

If there is no path from the given vertex through any datatype property, there is no solution for the search. The search algorithm terminates by knowing the reachable label sets of the vertex are all empty. Otherwise, the algorithm will find an optimal path with finite length, because h of a path with infinite length is infinitesimal due to the path length penalty. Most real world queries do not have cycles, so we prune cyclic paths during the search to further reduce the computation. This heuristic can be disabled for the applications with cyclic queries.

Phase 2) involves selecting a vertex as a source, and jointly finding multiple optimal paths that share a source. For each possible source class, we exploit the heuristic proposed in phase 1) to estimate the product of the highest path similarities of all paths as the score for the class. The algorithm in phase 1) runs using each class as the given source in descending order of the estimated score. If the real score of a class is greater than or equal to the estimated score of the remaining classes, those classes can be pruned. This algorithm also terminates with an optimal solution. The proof is similar to the proof of Theorem 1.

If the query graph has multiple sources, the algorithm in phase 2) runs for each source separately.

3.4 Query Reformulation

The primary focus of this paper is mapping. Thus, we only briefly explain the benefits of using q-mapping for query reformulation. A central challenge in query reformulation is missing mapping. In QODI, this challenge manifests as a mapping between a path in the query graph and a path in the source ontology graph. The determination of an ss-path correspondence anticipates that the paths may be of different lengths.

Given ss-path correspondences as mapping, the reformulation algorithm is simplified as traversing the mapped ss-paths, and generating a triple pattern for each graph edge. The URI of each edge in the ss-path is translated as the predicate of a triple. The subject and object of the triple are variables or literals assigned to the domain and range of the edge, respectively. Assigning variables to classes that are shared by multiple paths is an open research topic. We do not elaborate on this topic. For the query in Figure 2, a path correspondence and the resulting translated triple patterns are:

{Course,teacher,People,name,string} = {Course,offeredBy,Teacher,name,string}

?c1 course:offeredBy ?c2 . ?c2 teacher:name “Einstein” .

4 Experimental Setup

4.1 Test Sets

The test sets comprise three application domains: Life Science, Bibliography, and Conference Organization. The test cases include an ontology created by an international standards body, two ontologies created from direct mapping relational databases, and three ontologies used in OAEI [1].

The Life Science domain consists of Darwin Core and Specify. Darwin Core is an ontology at the center of the standardization efforts of the Global Biodiversity Information Foundation (GBIF), an organization concerned with cataloging the impacts of climate change. Darwin Core contains 18 classes, and 71 properties. The Specify ontology was created from direct mapping the SQL schema of the database in the Specify biological collections software package[4]. Specify is used to manage over 200 field specimen collections. The specify ontology has 11 classes, and 413 properties. The Bibliography domain comprises the UMBC ontology from OAEI, and an ontology that models DBLP, generated from the direct mapping of a relational database of DBLP metadata through Ultrawrap [14]. Class hierarchies are manually added. DBLP ontology has 17 classes, and 51 properties. The Conference domain consists of the two ontologies from OAEI, SIGKDD and SOFSEM. We have made the test suite available on our website[5].

Sets of test queries are also required, and were created as follows. First, groundtruth mappings were manually generated, containing all correct ss-path mappings between each pair of ontologies. Subsequently, a computer program systematically generated two kinds of SPARQL queries for each ontology. 1) A *PathOnly* query has a query graph consisting of only one ss-path in the groundtruth. 2) A *ClassAll* query has a query graph consisting of all ss-paths (at least two) that share a source in the groundtruth. A ClassAll query is the most complicated query with one conjunction over the source. In English specification, a PathOnly query asks for all values of a single attribute of a concept, and a ClassAll query asks for all values of all attributes of a concept. For ontology T in Figure 2, a PathOnly query could ask for names of all students taking courses, and a ClassAll query could ask for titles, time, and names of all students of all courses. Figure 4 shows examples of real PathOnly and ClassAll queries generated for the Specify ontology, as well as the meaning of both queries.

[4] `http://specifysoftware.org/`

[5] `http://ribs.csres.utexas.edu/qodi`

```
BASE <http://ribs.csres.utexas.edu/specify/>
Select ?v
Where {
 ?c0 <locality#Latitude1> ?v.
 ?c0 rdf:type <locality>.
}
```

(a) PathOnly query, asking for the latitude of all locations.

```
BASE <http://ribs.csres.utexas.edu/specify/>
Select ?v0 ?v1 ?v2 ?v3 ?v4 ?v5 ?v6
Where {
 ?c0 <determination#DeterminedDate> ?v0.
 ?c0 <determination#Remarks> ?v2.
 ?c1 <taxon#Name> ?v3.
 ?c2 <taxon#Name> ?v4.
 ?c3 <agent#DateOfBirth> ?v5
 ?c4 <agent#DateOfBirth> ?v6.
 ?c1 rdf:type <taxon>.
 ?c3 rdf:type <agent>.
 ?c0 <determination#Qualifier> ?v1.
 ?c0 <determination#ref-TaxonID> ?c1.
 ?c0 <determination#ref-PreferredTaxonID> ?c2.
 ?c0 <determination#ref-CreatedByAgentID> ?c3.
 ?c0 <determination#ref-ModifiedByAgentID> ?c4.
 ?c0 rdf:type <determination>.
 ?c2 rdf:type <taxon>.
 ?c4 rdf:type <agent>.
}
```

(b) ClassAll query, asking for the dates, remarks, and qualifiers of all determination of taxons, as well as the birthdays of the agents that determine the taxons.

Fig. 4. Real SPARQL queries generated for Specify ontology in the experiments

4.2 Baselines

We compare QODI against two kinds of baselines: ontology matching systems, and an ontology-based implementation of an existing relational data integration system.

For ontology matching baselines, a matcher computes the similarity between classes, object properties, and datatype properties. Given a query, each entity is translated to an entity in S with the highest similarity.

Clio is a relational data integration and exchange system that is closely related to QODI [7]. Clio generates mappings between attributes, and finds associations between those mappings through foreign key constraints. We implement baselines with similar ideas as Clio. A matcher first generates mappings between datatype properties by picking the ones with the highest similarity. Given a query, the baselines find the match candidates that contain all the mapped datatype properties. Clio asks a user to pick one match candidate, which is not allowed in our automated setting. We approximate this process by first picking the match candidates with highest similarity between source classes, and then picking the one with the least summation of path lengths.

We use three matchers for all methods. One matcher is substring string similarity that measures the portion of the longest common substrings between entity labels. The second matcher is SMOA string similarity between entity labels [15]. The third is AgreementMaker configured as detailed in OAEI 2010 conference track [5].

4.3 Metrics

The assessments are reminiscent of recall and precision used in ontology matching and information retrieval. $valid\_rate$ is the metric similar to recall, which is the proportion of queries with *complete q-mappings* generated, independent of correctness. We use $\#$ to represent *the number of*.

Definition 9 (complete q-mapping). *A q-mapping with a set of correspondences $\Omega_{Q,\bar{G}}$ is complete, if for every ss-path in the query graph Q, there exists a correspondence to an ss-path in $\bar{G}$ with non-zero confidence measure.*

$$valid\_rate = \frac{\#\text{ queries with complete q-mappings generated}}{\#\text{ queries}} \tag{7}$$

For measuring the precision of mapping systems, we consider the case that a query is correctly mapped, and also the case that a query is partially correctly mapped.

$$query\_precision = \frac{\#\text{ correctly mapped queries}}{\#\text{ queries}} \tag{8}$$

$$path\_precision = \frac{\sum_q \text{percentage of correctly mapped ss-paths in } q}{\#\text{ queries}} \tag{9}$$

A measure of ambiguity can facilitate the analysis of experimental results. An accurate measure of ambiguity is difficult, since it has to anticipate all possible application scenarios. We define an approximate measure of ambiguity, which only considers mapping between datatype properties as the source of ambiguity, and considers two datatype properties as mapped if a matcher assigns them the highest similarity.

Definition 10 (datatype ambiguous q-mapping). *Given a datatype property similarity measure S_D, a target ontology T, a source ontology S, a query q over T, and the set of ss-path correspondences Ω of q-mapping(T,S,q), the mapping is datatype ambiguous if for at least one ss-path correspondence $\pi_{p_t,p_s} \in \Omega$, $S_D(p_t, p_s) = \max_p S_D(p_t, p)$, and there exists a datatype property $d \notin p_s$, such that the similarity between d and the datatype property of p_t equals $S_D(p_t, p_s)$.*

$ambiguous\_rate$ is a measure of the proportion of queries that have datatype ambiguous q-mappings.

$$ambiguous\_rate = \frac{\#\text{ queries with datatype ambiguous q-mapping}}{\#\text{ queries}} \tag{10}$$

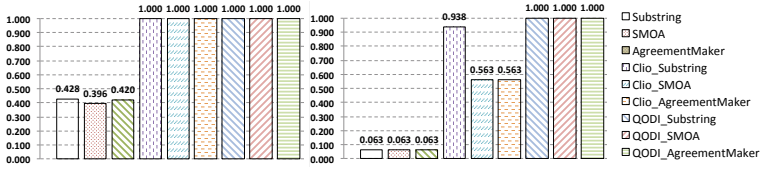
5 Experimental Results

Given a pair of ontologies, O_1 and O_2, the experiments are conducted on two directions of mappings: using O_1 as target and using O_2 as target. The results for the two mapping directions are shown separately for ambiguous_rate to distinguish the differences. For other metrics, the results are averaged. We set $\eta = 0.3$ based on the tuning on the Bibliography test set with PathOnly queries using Substring as matcher. Section 5.3 discusses the accuracy using different η. Due to the space limit, only part of the results are reporting. Please refer to the technical report for all results [16].

5.1 Valid_rate

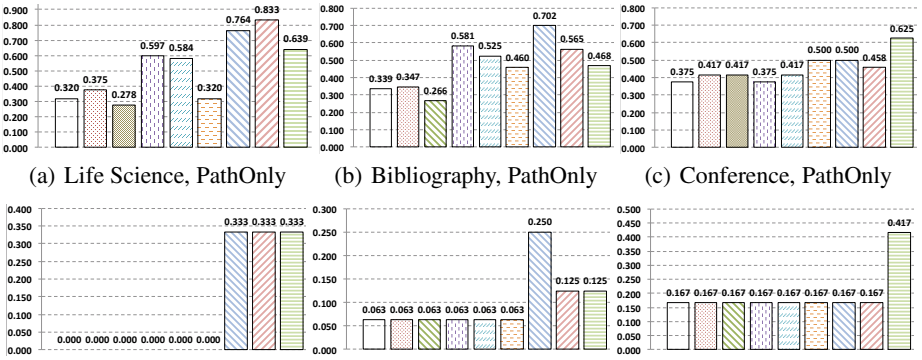
Figure 5 shows the valid_rate for Bibliography test set. Conference and Life Science test sets have similar numbers, and not reported here. The three methods of QODI achieve 100% valid_rate for all test sets. This is because QODI does not determine any entity mapping beforehand. Each path correspondence is assigned a confidence, and the mapped paths has the highest confidence.

The Clio baselines are able to generate complete mappings for the PathOnly query set but not all ClassAll queries. For some ClassAll queries, Clio cannot find a complete q-mapping if the mapped entities are incorrectly selected from ambiguous mappings.



(a) Bibliography, PathOnly (b) Bibliography, ClassAll (c) legend

Fig. 5. valid_rate for Bibliography test set. Higher number means better performance.



(a) Life Science, PathOnly (b) Bibliography, PathOnly (c) Conference, PathOnly

(d) Life Science, ClassAll (e) Bibliography, ClassAll (f) Conference, ClassAll

Fig. 6. query_precision for different test sets. Refer to Figure 5(c) for legend.

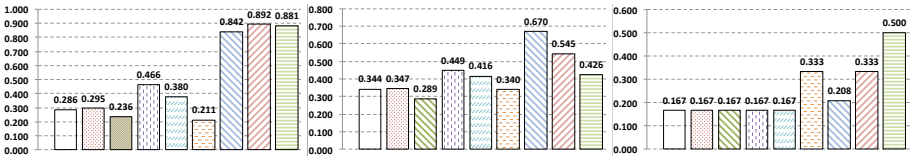
The comparison between QODI and Clio shows that disambiguation is important even for generating complete q-mappings regardless of correctness.

The ontology matching baselines are able to generate complete q-mappings for less than 50% of PathOnly queries, but barely generate complete q-mappings for ClassAll queries. The big gap between ontology matching baselines and Clio baselines demonstrates the importance of the missing mapping challenge.

5.2 Precision

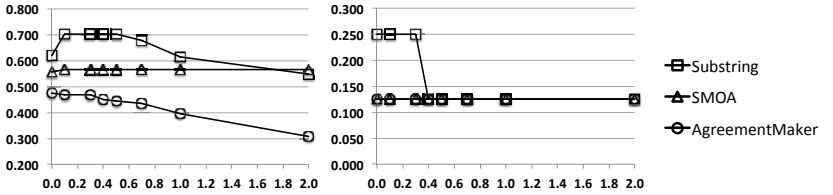
Figure 6 and 7 show the precisions of all methods. For all test sets, at least one QODI method dominates all baselines in terms of both precision measures. For ClassAll query sets, there are big gaps between QODI and all baselines. QODI is the only system that achieves non-zero query_precision for the Life Science test set with ClassAll query set. For ClassAll query set, each query has more than one path that shares a source. On one hand, more paths may lead to poor mapping results since each path may be mapped incorrectly. On the other hand the context from different paths may be used by QODI to map the correct source class shared by the paths. The precision results indicate the importance of resolving the ambiguous mapping challenge.

Comparing Clio with ontology matching baselines, for all test sets and all measures, at least one Clio baseline dominates or performs as well as ontology matching baselines.



(a) Life Science, ClassAll (b) Bibliography, ClassAll (c) Conference, ClassAll

Fig. 7. path_precision with ClassAll query sets for different test sets. Refer to Figure 5(c) for legend. path_precision for PathOnly query sets is the same as query_precision.



(a) Bibliography, PathOnly (b) Bibliography, ClassAll (c) legend

Fig. 8. query_precision of Bibliography test set when using different η (horizontal axis)

5.3 Parameter Tuning

Figure 8 shows the query_precision of Bibliography test set using different path length penalty parameter η. The results for Conference and Life Science sets are not reported due to space limit. With the same length difference, a large η gives big penalty. As a special case, $\eta = 0$ does not have any penalty on the path length.

For most cases, the penalty improves query_precision comparing to the case of $\eta = 0$. However, if η is too large, the query_precision can be decreased. With large η, the penalty of length dominates the similarities of source classes, datatype properties, and path labels in (2). Thus only the paths with the same lengths are considered as similar, ignoring the labels of the paths.

5.4 Ambiguity

As the primary motivation is the identification that mapping correctness may be query dependent (ambiguous), we assess how much of QODIs improved performance over Clio is explained by the presence of ambiguity and the respective systems ability to resolve it. In this section, we measure the ambiguous_rate of all test sets, and compute the query_precision of all methods over PathOnly queries with ambiguous mappings to measure the capability of disambiguation. If there is no ambiguity, the precision column is empty as shown in Table 1.

Two out of three test sets, Life Science and Bibliography, have non-zero ambiguous_rate. The ambiguous_rate measured with different matchers share similarities. All three matchers assert that Life Science with Darwin Core as target ontology has ambiguity, with rates from 0.139 to 0.333. Substring and SMOA agree on the ambiguity of

Table 1. The ambiguous_rate (row 2) and query_precision of queries with datatype ambiguous q-mapping (row 3, 4, 5) using SMOA as matcher. L↑ uses Darwin Core and L↓ uses Specify as the target ontology for the Life Science test set. B↑ uses UMBC and B↓ uses DBLP as the target ontology for the Bibliography test set. C↑ uses SIGKDD and C↓ uses SOFSEM as the target ontology for the Conference test set. If ambiguous_rate is zero, there is no query_precision for the queries with datatype ambiguous q-mapping. Higher query_precision means better performance.

	L↑	L↓	B↑	B↓	C↑	C↓
ambiguous_rate	0.194	0.000	0.177	0.000	0.000	0.000
SMOA	0.143	-	0.364	-	-	-
Clio_SMOA	0.429	-	0.364	-	-	-
QODI_SMOA	**0.714**	-	**0.636**	-	-	-

Bibliography with UMBC as target ontology, with rates 0.242 and 0.177. We only report the query_precision using SMOA as matcher in Table 1. Other matchers show similar results. For both L↑ and B↑, QODI achieves the highest query_precision on the queries with datatype ambiguous q-mappings. Comparing with Clio, the relative improvement of QODI is 66% and 75%. This shows that QODI is capable of disambiguation.

6 Related Work

Ontology matching has been well studied [2,6,3,8]. Many ontology matching systems compete in the OAEI [1], such as AgreementMaker [5] and RiMOM [12]. In the ontology matching world, most of systems focus on mapping between entities. In this paper, we define query-specific mapping for OBDI systems.

Clio, the state-of-the-art semi-automatic data integration and exchange system has close similarities with QODI [7]. Schema mapping in Clio is done in 2-steps: finding initial mappings between attributes; and associating mappings by logical inference through referential constraints. A semi-automatic OBDI system, Karma, is recently built to map structured data sources to ontologies [11]. For both Clio and Karma, the mapping is generated based on schemas alone. Neither system uses context from queries for resolving ambiguous mappings. Ontology based data access (OBDA) uses ontologies expressed in Description Logic as a conceptual view over data sources [4]. The mapping generated by QODI may be used for OBDA with proper representation.

7 Conclusions and Future Work

In this paper, we introduce query-specific ontology mapping, and implement an OBDI system, QODI. Departing from existing ontology matchers, QODI generates path correspondences, instead of entity correspondences, to facilitate query reformulation. The correspondences are discovered by heuristic search algorithm. A query is used as an input to the mapping to provide context for disambiguation and also reduce the mapping complexity.

Future work consists of at least three possible directions. First, the fundamental organization of QODI admits integration of user interaction for refinement. Second, path mappings can be accumulated over time as in pay-as-you-go systems. Third, new similarity measures and the relaxing of the basic assumptions can be explored.

Acknowledgements. This research is funded in part by the National Science Foundation grant IIS-1018554. Juan Sequeda was funded by the NSF Graduate Research Fellowship. We thank James Beach and Rodney Spears for providing a copy of the Specify database and sharing their knowledge of biodiversity informatics.

References

1. Ontology Alignment Evaluation Initiative, `http://oaei.ontologymatching.org/`
2. Bellahsene, Z., Bonifati, A., Rahm, E. (eds.): Schema Matching and Mapping. Springer (2011)
3. Bernstein, P., Madhavan, J., Rahm, E.: Generic schema matching, ten years later. In: Proc. VLDB, vol. 4(11) (2011)
4. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Poggi, A., Rodriguez-Muro, M., Rosati, R., Ruzzi, M., Savo, D.: The MASTRO system for ontology-based data access. In: Semantic Web, vol. 2(1), pp. 43–53 (2011)
5. Cruz, I., Antonelli, F., Stroe, C.: Agreementmaker: efficient matching for large real-world schemas and ontologies. In: Proc. VLDB, vol. 2(2), pp. 1586–1589 (2009)
6. Euzenat, J., Shvaiko, P.: Ontology matching. Springer-Verlag New York Inc. (2007)
7. Fagin, R., Haas, L.M., Hernández, M., Miller, R.J., Popa, L., Velegrakis, Y.: Clio: Schema mapping creation and data exchange. In: Borgida, A.T., Chaudhri, V.K., Giorgini, P., Yu, E.S. (eds.) Mylopoulos Festschrift. LNCS, vol. 5600, pp. 198–236. Springer, Heidelberg (2009)
8. Giunchiglia, F., Yatskevich, M., Shvaiko, P.: Semantic matching: Algorithms and implementation. In: Spaccapietra, S., et al. (eds.) Journal on Data Semantics IX. LNCS, vol. 4601, pp. 1–38. Springer, Heidelberg (2007)
9. Hart, P., Nilsson, N., Raphael, B.: A formal basis for the heuristic determination of minimum cost paths. IEEE Trans. Systems Science and Cybernetics 4(2), 100–107 (1968)
10. Heath, T., Bizer, C.: Linked data: Evolving the web into a global data space. Synthesis Lectures on the Semantic Web: Theory and Technology 1(1), 1–136 (2011)
11. Knoblock, C.A., et al.: Semi-automatically mapping structured sources into the semantic web. In: Simperl, E., Cimiano, P., Polleres, A., Corcho, O., Presutti, V. (eds.) ESWC 2012. LNCS, vol. 7295, pp. 375–390. Springer, Heidelberg (2012)
12. Li, J., Tang, J., Li, Y., Luo, Q.: Rimom: A dynamic multistrategy ontology alignment framework. IEEE Trans. Knowl. Data Eng. 21(8), 1218–1232 (2009)
13. Sequeda, J.F., Arenas, M., Miranker, D.P.: On directly mapping relational databases to RDF and OWL. In: Proc. WWW, Lyon, France, pp. 649–658 (2012)
14. Sequeda, J.F., Miranker, D.P.: Ultrawrap: SPARQL execution on relational data. Technical Report TR-12-10, University of Texas at Austin (2012)
15. Stoilos, G., Stamou, G., Kollias, S.D.: A string metric for ontology alignment. In: Gil, Y., Motta, E., Benjamins, V.R., Musen, M.A. (eds.) ISWC 2005. LNCS, vol. 3729, pp. 624–637. Springer, Heidelberg (2005)
16. Tian, A., Sequeda, J.F., Miranker, D.P.: QODI: Query as context in automatic data integration. Technical Report TR-12-15, University of Texas at Austin (2012)

TRank: Ranking Entity Types Using the Web of Data

Alberto Tonon[1], Michele Catasta[2], Gianluca Demartini[1], Philippe Cudré-Mauroux[1], and Karl Aberer[2]

[1] eXascale Infolab, University of Fribourg—Switzerland
firstname.lastname@unifr.ch

[2] EPFL, Lausanne—Switzerland
firstname.lastname@epfl.ch

Abstract. Much of Web search and browsing activity is today centered around entities. For this reason, Search Engine Result Pages (SERPs) increasingly contain information about the searched entities such as pictures, short summaries, related entities, and factual information. A key facet that is often displayed on the SERPs and that is instrumental for many applications is the entity *type*. However, an entity is usually not associated to a single generic type in the background knowledge bases but rather to a set of more specific types, which may be relevant or not given the document context. For example, one can find on the Linked Open Data cloud the fact that Tom Hanks is a person, an actor, and a person from Concord, California. All those types are correct but some may be too general to be interesting (e.g., person), while other may be interesting but already known to the user (e.g., actor), or may be irrelevant given the current browsing context (e.g., person from Concord, California). In this paper, we define the new task of ranking entity types given an entity and its context. We propose and evaluate new methods to find the most relevant entity type based on collection statistics and on the graph structure interconnecting entities and types. An extensive experimental evaluation over several document collections at different levels of granularity (e.g., sentences, paragraphs, etc.) and different type hierarchies (including DBPedia, Freebase, and schema.org) shows that hierarchy-based approaches provide more accurate results when picking entity types to be displayed to the end-user while still being highly scalable.

1 Introduction

Many online queries are about entities [14]. Commercial search engines are increasingly returning rich Search Engine Result Pages (SERPs) that contain not just ten blue links but also images, videos, news, etc. When searching for a specific entity, users may be presented in the SERP with a summary of the entity itself. This search task is known as ad-hoc object retrieval [20], that is, finding an entity described by a keyword query in a structured knowledge base. After correctly identifying the entity described by the user query, the subsequent task

H. Alani et al. (Eds.): ISWC 2013, Part I, LNCS 8218, pp. 640–656, 2013.

is that of deciding what entity information to present on the SERP among all potential pieces of information available in the knowledge base. It is possible, for example, to display pictures, a short textual description, and related entities.

One interesting entity facet that can be displayed in the SERP is its *type*. In public knowledge bases such as Freebase, entities are associated with several types. For example, the entity 'Peter Jackson' in Freebase[1] has 17 types, among which 'Person', 'Ontology Instance', 'Film director', and 'Chivalric Order Member' can be found. When deciding what to show on the SERP, it is important to select the few types the user would find relevant only. Some types are in most cases not compelling (e.g., 'Ontology Instance') while other types (e.g., 'Film director') may be interesting for a user who does not know much about the entity. Users who already know the entity but are looking for some of its specific facets might be interested in less obvious types (e.g., 'Chivalric Order Member', and its associated search results).

More than just for search, entity types can be displayed to Web users while browsing and reading Web pages. In such a case, pop-ups displaying contextual entity summaries (similar to the ones displayed on SERPs like in Google's Knowledge Panel) can be shown to the users who want to know more about a given entity she is reading about. In this case again, picking the types that are relevant is critical and highly context-dependent.

A third example scenario is to use selected entity types to summarize the content of Web pages or online articles. For example, one might build a summary for a given news article by extracting the most important entities in the article and listing their most relevant types (e.g., 'this article is about two actors and the president of Kenya').

In this paper, we focus on the novel task of ranking available entity types based on their relevance given a context. We propose several methods exploiting the entity type hierarchy (i.e., types and their subtypes like 'person' and 'politician'), collection statistics such as the popularity of the types or their co-occurrences, and the graph structure connecting semantically related entities (potentially through the type hierarchy).

We experimentally evaluate our different approaches using crowdsourced judgments on real data and extracting different contexts (e.g., word only, sentence, paragraph) for the entities. Our experimental results show that approaches based on the type hierarchy perform more effectively in selecting the entity types to be displayed to the user. The combination of the proposed ranking functions by means of learning to rank models yields the best effectiveness. We also assess the scalability of our approach by designing and evaluating a Map/Reduce version of our system, *TRank*, over a large sample of the CommonCrawl dataset[2] containing `schema.org` annotations.

In summary, the main contributions of this paper are:

- The definition of the new task of entity type ranking, whose goal is to select the most relevant types for an entity given some context.

[1] `http://www.freebase.com/edit/topic/en/peter_jackson`
[2] `http://commoncrawl.org/`

- Several type-hierarchy and graph-based approaches that exploit both schema and instance relations to select the most relevant entity types based on a query entity and the user browsing context.
- An extensive experimental evaluation of the proposed entity type ranking techniques over a Web collection and over different entity type hierarchies including YAGO [23] and DBpedia [1] by means of crowdsourcing relevance judgements.
- A scalable version of our type ranking approach evaluated over a large annotated Web crawl.

The rest of the paper is structured as follows. We start below by describing related work surveying entity-search and ad-hoc object retrieval techniques. We formally define our new type ranking task in Section 3 and propose a series of approaches to solve it based on collection statistics, type hierarchies, and entity graphs in Section 4. Section 5 presents experimental results comparing the effectiveness of our various entity ranking approaches over different document collections and type hierarchies as well as a scalability validation of our Map/Reduce implementation over a large corpus. Finally, we conclude in Section 6.

2 Related Work

Entity-centric data management is an emerging area of research at the intersection of several fields including Databases, Information Retrieval, and the Semantic Web. In this paper we target the specific problem of assigning types to entities that have been extracted from a Web page and correctly identified in a preexisting knowledge base.

Classic approaches to Named Entity Recognition (NER) typically provide as output some type information about the identified entities; In most cases, such types consist of a very limited set of entities including Person, Location, and Organization (see e.g., [4,5]). While this is useful for applications that need to focus on one of those generic types, for other applications such as entity-based faceted search it would be much more valuable to provide specific types that are also relevant to the user's browsing context.

In the field of Information Retrieval, entity retrieval has been studied for a few years. In this context, TREC[3] organized an Entity Track where different entity-centric search tasks have been studied: Four entity types were considered in that context, i.e., people, products, organizations, and locations. Type information can also be used for entity search tasks, e.g., by matching the types of the entities in the query to the types of the retrieved entities (see for instance [7]). In the NLP field, entity extraction methods are continuously being developed. Here also, the types that are considered are typically rather limited. For example, in the method proposed in [9] 18 types are considered. In [19,18], authors propose a NER system to recognize 100 entity types using a supervised approach.

[3] `http://trec.nist.gov`

The starting point to define the 100 entity types is the BBN linguistic collection[4] which includes 12 top types and 64 subtypes.

The Semantic Web community has been creating large-scale knowledge bases defining a multitude of entity types. Efforts such as YAGO [23] have assigned to LOD entities many types by combining Wikipedia categories and WordNet senses. More recently, projects such as DBpedia [1] and Freebase [2] have collected large collections of structured representations of entities along with their related types. Such knowledge bases hence represent extremely valuable resources when working on entity type ranking as we do in this paper.

In a recent demo [25], the task of selecting the most relevant types to be used to summarize an entity has been proposed. However, the focus of this work was on generating an entity description of a given size, while our focus is to select the most relevant types given the context in which the entity is described. Similarly to that work, we build our approaches using large knowledge bases such as YAGO and DBpedia. Another related approach is *Tipalo* [10], where the authors propose an algorithm to extract entity types based on the natural language description of the entity taken from Wikipedia.

Several applications of our techniques could be based on existing work. For instance, entity-type ranking could be applied on open-domain Question Answering [13], where candidate answers are first generated and later on filtered based on the expected answer types. For systems like Watson [26], identifying specific and relevant entity types could potentially significantly improve effectiveness. Another application depending on high-quality entity types is entity resolution over datasets of different entity types. In [27], the authors evaluate their approach on top of four entity types (that is, persons, addresses, schools, and jobs). The availability of more specific entity types would probably be beneficial for this type of task as well.

3 Task Definition

Given a knowledge base containing semi-structured descriptions of entities and their types, we define the task of *entity type ranking* for a given entity e appearing in a document d as the task of ranking all the types $T_e = \{t_1, \ldots, t_n\}$ associated to e based on their relevance to its textual context c_e from d. In RDFS/OWL, the set T_e is typically given by the objects that are related to the URI of e via the `<rdfs:type>` predicate. Moreover, we take into consideration entities connected to e via a `<owl:sameAs>` to URIs of other selected ontologies and we add to T_e all the types directly attached to them. For example, `<dbpedia:Tom_Cruise>` has an `<owl:sameAs>` connection to `<freebase:Tom_Cruise>`, which allows us to add the new type `<freebase:fashion_models>`.

The context c_e of an entity e is defined as the textual content surrounding the entity taken from the document d in which e is mentioned. This context can have a direct influence on the rankings. For example, the entity 'Barack Obama' can be mentioned in a Gulf War context or in a golf tournament context.

[4] `http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC2005T33`

The most relevant type for 'Barack Obama' is probably different given one or the other context. The different context types we consider in this paper are: i) three paragraphs around the entity reference (one paragraph preceding, one following, and the paragraph containing the entity); ii) one paragraph only, containing the entity mention; iii) the sentence containing the entity reference; and iv) the entity mention itself with no further textual context.

To rank the types by their relevance given a context, we exploit hierarchies of entity types. In RDFS/OWL, a type hierarchy is typically defined based on the predicate `<rdfs:subClassOf>`. For example, in DBpedia we observe that `<dbpedia-owl:Politician>` is a subclass of `<dbpedia-owl:Person>`. Knowing the relations among types and their depth in the hierarchy is often helpful when automatically ranking entity types. For example, given a type hierarchy related to a specific entity, we might prefer a more specific type rather than a too general one.

We evaluate the quality of a given ranking $(t_i, \ldots, t_j)$ by using ground truth relevance judgements assessing which types are most relevant to an entity e given a context c_e. We discuss rank-based evaluation metrics in Section 5.

4 Approaches to Entity Type Ranking

4.1 *TRank* System Architecture

Our solution, *TRank*, automatically selects the most appropriate entity types for an entity given its context and type information. *TRank* implements several components to extract entities and automatically determine relevant types. First, given a Web page (e.g., a news article), we identify entities mentioned in the textual content of the document using state-of-the-art NER focusing on persons, locations, and organizations.[5] Next, we use an inverted index constructed over DBpedia literals attached to its URIs and use the extracted entity as a query to the index to select the best-matching URI for that entity.[6] Then, given an entity URI, we retrieve (for example, thanks to a SPARQL query to a knowledge base) all the types attached to the entity. In this way, we obtain types such as `<owl:Thing>`, `<yago:JapanPrizeLaureates>` and `<yago:ComputerPioneers>` for the entity `<dbpedia:Tim_Berners-Lee>`. Finally, our system produces a ranking of the resulting types based on the textual context where the entity has been mentioned. A summary of the different steps involved is depicted by Figure 1.

Integrating Different Type Hierarchies. For the purpose of our task, we require a large, integrated collection of entity types to enable fine-grained typing of entities. There are several large ontologies available, both manually constructed [16]

[5] The current implementation of our system adopts a Conditional Random Field approach to identify entities [8].

[6] This is the same baseline approach used in [6] and in [24] for Entity Linking.

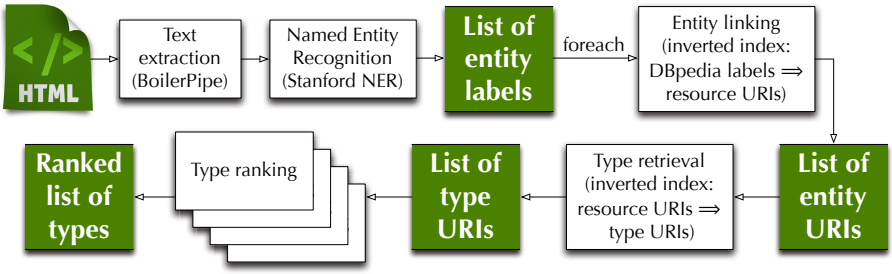


Fig. 1. The *TRank* Architecture

as well as based on the widespread success of Wikipedia combined with information extraction algorithms [1,23]. However, the lack of alignments among such ontologies hinders the ability of comparing types belonging to different collections.

In *TRank*, we exploit pre-existing mappings provided by DBpedia and PARIS [22] to build a coherent tree of 447,260 types, rooted on `<owl:Thing>` and with a depth of 19. The tree is formed by all the `<rdfs:subClassOf>` relationships among DBpedia, YAGO and schema.org types. To eliminate cycles and to enhance coverage, we exploit `<owl:equivalentClass>` to create `<rdfs:subClassOf>` edges pointing to the parent class (in case one of the two Classes does not have a direct parent). Considering that the probabilistic approach employed by PARIS does not provide a complete mapping between DBpedia and YAGO types, we have manually added 4 `<rdfs:subClassOf>` relationships (reviewed by domain experts) to obtain a single type tree rather than a forest of 5 trees.[7] Figure 2 shows a visual representation of the integrated type hierarchy used by *TRank*.

Entity Type Retrieval and Ranking. Finally, given the entity URI we retrieve all its types (from a background RDF corpus or from a previously created inverted index) and rank them given a context. In this paper, we use the Sindice-2011 RDF dataset[8] [3] to retrieve the types, which consists of about 11 billion RDF triples.

The proposed approaches for entity type ranking can be grouped in entity-centric, context-aware, and hierarchy-based. Figure 3 shows on which data such approaches are based. The entity-centric approaches look at the relation of the entity e with other entities in a background knowledge base following edges such as `<dbpedia-prop:wikiLink>` and `<owl:sameAs>`. Context-aware approaches exploit the co-occurrence of the entity e with other entities in the same textual

[7] The type hierarchy created in this way is available in the form of a small inverted index that provides for each type the path to the root and its depth in the hierarchy at `http://exascale.info/TRank`

[8] `http://data.sindice.com/trec2011/`

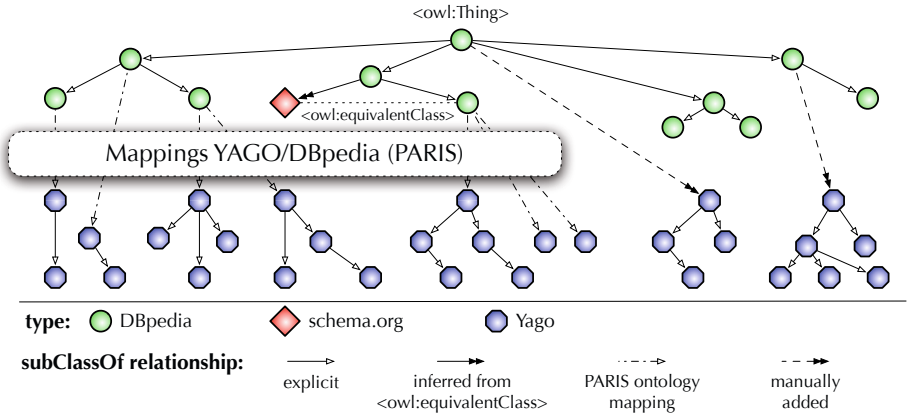


Fig. 2. The integrated type hierarchy

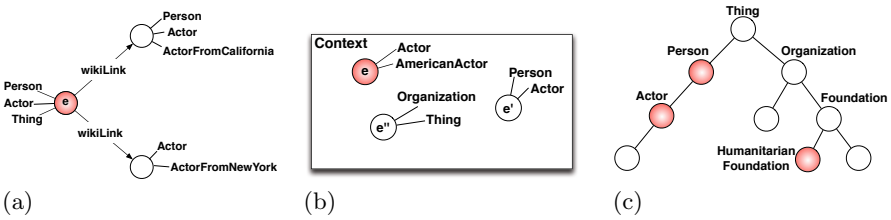


Fig. 3. (a) Entity-centric (b) Context-aware (c) Hierarchy-based type ranking

context. Hierarchy-based approaches look at the structure of the type hierarchy and rank types based on it.

4.2 Entity-Centric Ranking Approaches

We now turn to the detailed description of several techniques to rank entity types. The first group of approaches we describe only considers background information about a given entity and its types without taking into account the context in which the entity appears.

Our first basic approach (FREQ) to rank entity types is based solely on the frequency of those types in the background knowledge base ranking first the most frequent type of an entity. For example, the type Person has a higher frequency (and thus is more popular) than EnglishBlogger.

Our second approach (WIKILINK) exploits the relations existing between the given entity and further entities in the background knowledge base. Hence, we count the number of neighboring entities that share the same type. This can be performed by issuing the following SPARQL queries retrieving connected entities from/to e to rank $t_i \in T_e$:

```
SELECT ?x WHERE { <e> <dbpedia-prop:wikilink> ?x . ?x <rdfs:type> <t_i> }
SELECT ?x WHERE { ?x <dbpedia-prop:wikilink> <e> . ?x <rdfs:type> <t_i> }
```

For example, in Figure 3a to rank types for the entity e we exploit the fact that linked entities have also the type 'Actor' to rank it first.

In a similar way, we exploit the entity graph from the knowledge base by following `<owl:sameAs>` connections and observing the types attached to such URIs (SAMEAS):

```
SELECT ?x WHERE {<e> <owl:sameAs> ?x . ?x <rdfs:type> <t_i> }
```

Our next approach (LABEL) adopts text similarity methods. We consider the label of e and measure its TF-IDF similarity with other labels appearing in the background knowledge base in order to find related entities.[9] At this point, we inspect the types of the most related entities to rank the types of e. More specifically, we select the top-10 entities having the most similar labels to e and rank types based on the frequency of $t_i \in T_e$ for those entities.

4.3 Context-Aware Ranking Approaches

We describe approaches leveraging the entity context below. A first approach (SAMETYPE) taking into account the context c_e in which e appears is based on counting how many times each type $t_i \in T_e$ appears in the co-occurring entities $e' \in c_e$ also mentioned in the context. In this case, we consider a match whenever the same type URI is used by e and e', or when the type of e' has the same label as the type from e. For example, in Figure 3b we rank first the type 'Actor' for the entity e because it co-occurs with other entities of type Actor in the same context.

A slightly more complex approach (PATH) leverages both the type hierarchy and the context in which e appears. Given all entities appearing in the context $e' \in c_e$, the approach measures how similar the types are based on the type hierarchy. We measure the degree of similarity by taking the intersection between the paths from the root of the type hierarchy (i.e., `<owl:Thing>`) to $t_i \in T_e$ and to $t_j \in T_{e'}$. For instance, when ranking types for the entity 'Tom Hanks' in a context where also 'Tom Cruise' appears, we measure the similarity between the types by considering the common paths between the root of the type hierarchy and both types, e.g., "Thing-Agent-Person-Artist-Actor-AmericanTelevisionActors" and "Thing-Agent-Person-Artist-Actor-ActorsFromNewJersey" would be considered as highly similar. On the other hand, the 'Tom Hanks' type path "Thing-PhysicalEntity-CausalAgent-Person-Intellectual-Scholar-Alumnus-CaliforniaStateUniversity,SacramentoAlumni" is not very similar with the previous 'Tom Cruise' path. Hence, the approach ranks the 'AmericanTelevisionActors' type higher given the context in which it appears.

4.4 Hierarchy-Based Ranking Approaches

The more complex techniques described below make use of the type hierarchy and measure the depth of an entity type t_i attached to e in order to assess its

[9] This can be efficiently performed by means of an inverted index over entity labels.

relevance. We define the DEPTH ranking score of a type t_i as the depth of t_i in the type hierarchy. This approach favors types that are more specific (i.e., deeper in the type hierarchy).

In some cases, the depth of an entity type in the hierarchy may not be enough. To detect the most relevant entity types, it might also be useful to determine the branch in the type hierarchy where the most compelling entity types are defined. In that context, we define a method (ANCESTORS) that takes into consideration how many ancestors of $t_i \in T_e$ are also a type of e. That is, if $Ancestors(t_i)$ is the set of ancestors of t_i in the integrated type hierarchy, then we define the score of t_i as the size of the set $\{t_j | t_j \in Ancestors(t_i) \wedge t_j \in T_e\}$. For example, in Figure 3c we rank first the type 'Actor' because 'Person' is its ancestor and it is also a type of e. On the other hand, the type 'Humanitarian Foundation' has a bigger depth but no ancestor which is also a type of e.

A variant of this approach (ANC_DEPTH) considers not just the number of such ancestors of t_i but also their depth. Thus,

$$ANC\_DEPTH(t_i) = \sum_{t_j \in Ancestor(t_i) \wedge t_j \in T_e} depth(t_j). \quad (1)$$

4.5 Learning to Rank Entity Types

Since *TRank* ranking approaches cover fairly different types of evidence (based on the entity-graph, the context, or the type hierarchy) to assess the relevance of a type, we also propose to combine our different techniques by determining the best potential combinations using a training set, as it is commonly carried out by commercial search engines to decide how to rank Web pages (see for example [15]). Specifically, we use decision trees [11] and linear regression models to combine the ranking techniques described above into new ranking functions. The decision tree method we used is M5 [21], which is specifically designed for regression problems. The effectiveness of this approach is discussed in Section 5.

4.6 Scalable Entity Type Ranking with MapReduce

Ranking types using the above methods for all the entities identified in a large-scale corpus using a single machine and SPARQL end-points is impractical, given the latency introduced by the end-point and the intrinsic performance limitations of a single node. Instead, we propose a self-sufficient and scalable Map/Reduce architecture for *TRank*, which does not require to query any SPARQL end-point and which pre-computes and distributes inverted indices across the worker nodes to guarantee fast lookups and ranking of entity types. More specifically, we build an inverted index over the DBpedia 3.8 entity labels for the entity linking step and an inverted index over the integrated *TRank* type hierarchy which provides, for each type URI, its depth in the integrated type hierarchy and the path to the root of the hierarchy. This enables a fast computation of the hierarchy-based type ranking methods proposed in Section 4.4.

5 Experiments

5.1 Experimental Setting

We created a ground truth of entity types mentioned in 128 news articles selected from the top news of each category from the New York Times (NYT) website during the Feb 21 – Mar 7 2013 period. On average, each article contains 12 entities. After the entity linking step, each entity gets associated with an average of 10.2 types from our Linked Data collection. We crowdsourced the selection of the most relevant types by asking the workers to select the most relevant type given a specific textual context.

Crowdsourced Relevance Judgements. We used paid crowdsourcing to create the ground truth.[10] We decided to ask anonymous Web users rather than creating the ground truth ourself as they are a real sample of Web user who could benefit from the envisioned application. Each task, which was assigned to 3 different workers from the US, consists of asking the most relevant type for 5 different entities, and was paid $0.10 for entities without context and $0.15 for entities with a context. Additionally, we allowed the workers to tag entities that were wrongly extracted, and to add an additional type if the proposed ones were not satisfactory. Overall, the relevance judgement creation cost $190.

In order to better understand how to obtain the right information from the crowd, we ran a pilot study where we compared different task designs for the entity type judgement task. We assessed the approach of asking the crowd to select all types which are relevant for an entity given its context as compared to asking which is the best type. Given the results of the pilot study, we selected the design that asks the worker to pick the *best* type only as this also best models the use case of showing one single entity type to a user browsing the Web. To generate our ground truth out of the crowdsourcing results, we consider as relevant each type which has been selected by at least one worker, in order to obtain binary judgements. We take the number of workers who selected a type as its relevance score in a graded relevance setting.

Evaluation Measures. As the main evaluation measures for comparing different entity type ranking methods, we use Mean Average Precision (MAP). Average Precision (AP) for the types T_e of an entity e is defined as

$$AP(T_e) = \frac{\sum_{t_i \in T_e} rel(t_i) \cdot P@i}{|Rel(T_e)|} \tag{2}$$

where $rel(t_i)$ is 1 if t_i is a relevant type for the entity e and 0 otherwise, $Rel(T_e)$ is the set of relevant types for e, and $P@i$ indicates Precision at cutoff i. MAP is defined as the mean of AP over all entities in the collection.

MAP is a standard evaluation measure for ranking tasks which consider binary relevance: A type t_i is either correct or wrong for an entity e. Since the

[10] We run our tasks over the Amazon MTurk platform. The collected data and task designs are available for others to reuse at `http://exascale.info/TRank`

original relevance judgements are not binary (i.e., more than one worker can vote for a type and thus have a higher relevance value than a type with just one vote), we also measure the Normalize Discounted Cumulative Gain (NDCG) [12], which is a standard evaluation measure for ranking tasks with non-binary relevance judgements. NDCG is defined based on a gain vector G, that is, a vector containing the relevance judgements at each rank. Then, the *discounted cumulative gain* measures the overall gain obtained by reaching rank k putting more weight at the top of the ranking: $DCG[k] = \sum_{j=1}^{k} G[j]/(log_2(1+j))$. To compute the final NDCG, we normalize it by dividing DCG by its optimal value obtained with the optimal gain vector which puts the most relevant results first.

5.2 Dataset Analysis

Out of the NYT articles we have crawled, we created four different datasets to evaluate and compare our approaches for the entity type ranking task. First, we use a collection consisting exclusively of entities and their types as extracted from the news articles. This collection is composed of 770 distinct entities: out of the original 990 extracted entities we consider only those with at least two types and we removed the errors in NER and entity linking which were identified by the crowd during the relevance judgements.

Sentence Collection. We built a Sentence collection consisting of all the sentences containing at least two entities. In this and the following collections we asked the human assessor to judge the relevance of a type given a context (e.g., a sentence). This collection contains 419 context elements composed of an average number of 32 words and 2.45 entities each.

Paragraph Collection. We constructed a collection consisting of all the paragraphs longer than one sentence and containing at least two entities having more than two types. This collection contains 339 context elements composed of an average number of 66 words and 2.72 entities each.

3-Paragraphs Collection. The last collection we have constructed contains the largest context for an entity: the paragraph where it appears together with the preceding and following paragraphs in the news article. This collection contains 339 context elements which are composed on average of 165 words each. The entire context contains on average 11.8 entities which support the relevance of the entities appearing in the central paragraph.

5.3 Experimental Results

Table 1 shows the overall effectiveness obtained by the proposed approaches. When we compare the results obtained among the different collections (i.e., entity-only, sentence, paragraph, and 3 paragraphs) we observe that effectiveness values obtained without context are generally higher, supporting the conclusion that the type ranking task for an entity without context is somehow easier than when we need to consider the story in which it is mentioned. Among the entity

Table 1. Type ranking effectiveness for different textual contexts. Statistically significative improvements (t-test $p < 0.05$) of the regression methods over the best ranking approach are marked with * and of the best hierarchy-based method over the best method from the other groups with †.

	Entity-only		Sentence		Paragraph		3-Paragraphs	
Approach	**NDCG**	**MAP**	**NDCG**	**MAP**	**NDCG**	**MAP**	**NDCG**	**MAP**
FREQ	0.6284	0.4659	0.5409	0.3758	0.5315	0.3739	0.5250	0.3577
WIKILINK-OUT	0.6874	0.5406	0.6050	0.4521	0.6063	0.4550	0.6059	0.4444
WIKILINK-IN	0.6832	0.5342	0.5907	0.4213	0.5879	0.4254	0.5853	0.4143
SAMEAS	0.6848	0.5328	0.6049	0.4310	0.5990	0.4221	0.6172	0.4417
LABEL	0.6672	0.5067	0.6075	0.4265	0.5883	0.4104	0.5821	0.4034
SAMETYPE	-	-	0.6024	0.4452	0.5917	0.4327	0.5813	0.4256
PATH	-	-	0.6507	0.4956	0.6538	0.4974	0.6315	0.4742
DEPTH	0.7432	0.6128	0.6754	0.5385	0.6797	0.5475	0.6741	0.5354
ANCESTORS	0.7424	0.6154	**0.6967**†	**0.5637**†	**0.6949**†	**0.5662**†	**0.6879**†	**0.5562**†
ANC_DEPTH	**0.7469**†	**0.6236**†	0.6832	0.5488	0.6885	0.5546	0.6796	0.5423
DEC-TREE	**0.7614**	**0.6361**	**0.7373***	**0.6079***	**0.7979***	**0.7019***	**0.7943***	**0.6914***
LIN-REG	0.7373	0.6079	0.6906	0.5579	0.6987	0.5702	0.6899	0.5529

centric approaches, in most of the cases the best approach is WIKILINK-OUT, that is, the approach that follows the `<dbpedia-prop:wikiLink>` edges starting from the entity e and that checks the frequency of its types among its connected entities. Among the context-aware approaches, the PATH method performs best. Interestingly, the hierarchy-based approaches clearly outperform the methods looking at the context or at the entity itself. The relatively simple DEPTH approach performs very effectively. The approaches looking at the ancestor of a type in the integrated hierarchy are the most effective approaches for ranking entity types among the ones we propose. Nevertheless, there are cases in which context-aware approaches rank types better than hierarchy-based ones. For example, in some document "Mali" co-occurs with "Paris", "Greece", and "Europe". The top-3 results selected by ANCESTORS for "Mali" are "LeastDevelopedCountries", "LandlockedCountries", and "French-speakingCountries", which are all non-relevant since they are too specific with respect to the context. In contrast, the top-3 types selected by PATH: "PopulatedPlace", "Place", and "Country", are all relevant according to the crowd.

To evaluate the combination of approaches using machine-learning methods, we run 10-fold cross validation over 7884, 11875, 11279, and 11240 data points in the four different collections. Out of the ranking approaches we have proposed, we selected 12 features which cover all the different methodologies (i.e., entity-centric, context-aware, and hierarchy-based) to train regression models for entity type ranking. We observe that the best performing method is the one based on decision trees, which significantly outperforms all other approaches.

Figure 4 shows the evolution of MAP and NDCG values by looking at entities with a different number of associated types. Generally speaking, we see that entity having many different types are more difficult to handle. On the other hand, even for the simple approach FREQ, when few types are assigned to an entity we obtain effective results. On the right side of Figure 4, we can observe the robustness of DEC-TREE over entities with an increasing number of types.

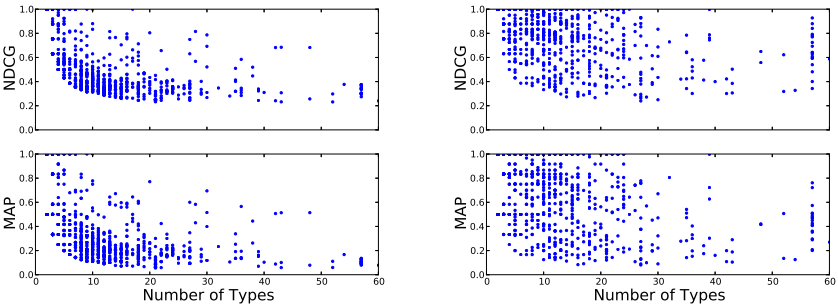


Fig. 4. MAP and NDCG of FREQ (left) and DEC-TREE (right) for entities with different numbers of types on the 3-paragraphs collection

Crowd-powered Entity Type Assignment. In some cases the knowledge base may not contain types that are good enough. For example, some entities have only `<owl:Thing>` and `<rdfs:Resource>` attached to them. In such cases, we asked the crowd to suggest a new type for the entity they are judging. While extend existing LOD ontologies with additional schema element is not the focus of this paper, we observe that this can be easily achieved by means of crowdsourcing. The suggestion of new types from the crowd may also suggest an error at the entity linking step (i.e., a wrong URI has been assigned to the entity mention). Some examples of crowd-originated entity types are shown in Table 2.

TRank *Scalability.* We ran the MapReduce *TRank* pipeline over a sample of CommonCrawl containing `schema.org` annotations. Upon writing this paper, CommonCrawl is formed by 177 valid crawling segments, accounting for 71TB of compressed Web content. We uniformly sampled 1TB of data over the 177 segments, and kept only the HTML content with `schema.org` annotations. This resulted in a corpus of 1,310,459 HTML pages, for a total of 23GB (compressed). Our MapReduce testbed is composed of 8 slave servers, each with 12 cores at 2.33GHz, 32GB of RAM and 3 SATA disks. The relatively small size of the 3 Lucene inverted indices ($\sim$ 600MB) used by the *TRank* pipeline allowed us to replicate the indices on each single server (transparently via HDFS). In this way, no server represented a read hot-spot or, even worse, a single point of failure.

Table 2. Examples of crowd-originated entity types

Entity Label	Existing Types	Crowd Suggested Type
David Glassberg	Alumnus, Resource, Northwestern University Alumni, American television journalists	New York City policeman
Fox	Thing, Eukaryote	Television Network
Bowie	Minor league team, Minor league sports team	Musical Artist
Atlantic	Resource, Populated Place	Ocean
European Commission	Type of profession, Landmark	Governmental Organizations
Childress	Thing, Resource	Locality

Table 3. Efficiency breakdown of the *TRank* MapReduce pipeline

Text Extraction	NER	Entity Linking	Type Retrieval	Type Ranking
18.9%	35.6%	29.5%	9.8%	6.2%

Table 4. CommonCrawl sample statistics

Domain	% in corpus
youtube.com	39.65
blogspot.com	9.26
over-blog.com	0.67
rhapsody.com	0.54
fotolog.com	0.52

Schema.org type	% in corpus
http://schema.org/VideoObject	40.79
http://schema.org/Product	32.66
http://schema.org/Offer	28.92
http://schema.org/Person	20.95
http://schema.org/BlogPosting	18.97

We argue that the good performance of our MapReduce pipeline is in majorly due to the use of small, pre-computed inverted indices instead of expensive SPARQL queries.

Processing the corpus on such a testbed takes 25 minutes on average, that is, each server runs the whole *TRank* pipeline at 72 documents per second. Table 3 shows a performance breakdown for each component of the pipeline. The value reported for "Type Ranking" refers to the implementation of ANCESTORS, but it is comparable for all the other techniques presented in the paper (except the ones based on the Learning to Rank approach, which we did not test in MapReduce).

The observed `schema.org` class distributions almost overlaps with the one previously found by [17] (see Table 4).[11] Table 5 shows the most frequent entity types selected by *TRank* for entities contained in our sample of CommonCrawl. We can observe how *TRank* types refer to specific entities mentioned in topic-specific pages as, for example, `<yago:InternetCompaniesOfTheUnitedStates>` entities that are contained in `<http://schema.org/Product>` Web pages.

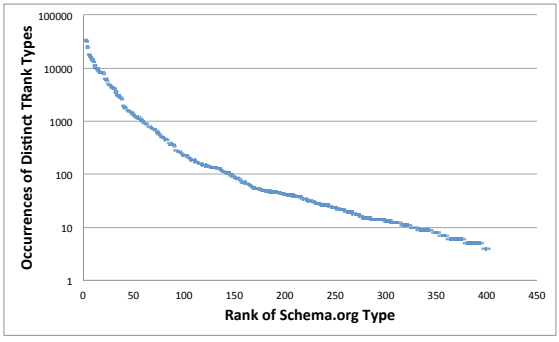


Fig. 5. Occurrences of distinct *TRank* types in CommonCrawl (log scale)

Figure 5 shows the variety of entity types selected by *TRank* for Web pages annotated with different `schema.org` classes. We clearly recognize a power-low

[11] More statistics can be found at `http://exascale.info/TRank`

Table 5. Co-occurrences of top Schema.org annotations with entity types.

Schema.org type	top-3 most frequent *TRank* types
http://schema.org/VideoObject	<dbpedia-owl:GivenName> <dbpedia-owl:Settlement> <dbpedia-owl:Company>
http://schema.org/Product	<yago:InternetCompaniesOfTheUnitedStates> <yago:PriceComparisonServices> <dbpedia-owl:Settlement>
http://schema.org/Offer	<yago:InternetCompaniesOfTheUnitedStates> <yago:PriceComparisonServices> <dbpedia-owl:Company>
http://schema.org/Person	<dbpedia-owl:GivenName> <dbpedia-owl:Company> <yago:FemalePornographicFilmActors>

distribution where the top `schema.org` classes contain very many different entity types while most of the others have a low diversity of entity types.

6 Conclusions

In this paper, we focused on the new task of ranking types for online entities given some textual context and links to background knowledge bases. Numerous applications can be developed once the most relevant entity types are correctly determined, including SERP enrichment, faceted browsing, and document summarization. We proposed different classes of ranking approaches and evaluated their effectiveness using crowdsourced relevance judgments. We also evaluated the efficiency of the proposed approach by taking advantage of inverted indices for fast access to entity and type hierarchy information and of a MapReduce pipeline for efficient entity type ranking over a Web crawl.

Our experimental results show that the approaches considering the relations between entity types in the overall type hierarchy outperform the other classes of approaches. A regression model learned over training data combining the different classes of ranking approaches significantly outperforms the individual ranking approaches, reaching a Mean Average Precision value of 0.70. As future work, we aim at improving *TRank* effectiveness by differentiating types and roles to design new ranking approaches based both on natural types and on the interaction among entities in the context. In addition, we plan to test the behavior of our system with different domains and ontologies, and to evaluate the user impact of entity typing by running a large-scale experiment using a browser plugin to display contextual entity types while the user is surfing.

Acknowledgments. This work was supported by the Swiss National Science Foundation under grant number PP00P2_128459, and by the Haslerstiftung in the context of the Smart World 11005 (Mem0r1es) project.

References

1. Bizer, C., Lehmann, J., Kobilarov, G., Auer, S., Becker, C., Cyganiak, R., Hellmann, S.: DBpedia - A crystallization point for the Web of Data. Journal of Web Semantics, 154–165 (2009)

2. Bollacker, K., Evans, C., Paritosh, P., Sturge, T., Taylor, J.: Freebase: a collaboratively created graph database for structuring human knowledge. In: SIGMOD, pp. 1247–1250 (2008)
3. Campinas, S., Ceccarelli, D., Perry, T.E., Delbru, R., Balog, K., Tummarello, G.: The Sindice-2011 dataset for entity-oriented search in the web of data. In: 1st International Workshop on Entity-Oriented Search (EOS), pp. 26–32 (2011)
4. Ciaramita, M., Altun, Y.: Broad-coverage sense disambiguation and information extraction with a supersense sequence tagger. In: EMNLP, pp. 594–602 (2006)
5. Cunningham, H., Humphreys, K., Gaizauskas, R., Wilks, Y.: GATE: a general architecture for text engineering. In: ANLC, pp. 29–30 (1997)
6. Demartini, G., Difallah, D.E., Cudré-Mauroux, P.: ZenCrowd: leveraging probabilistic reasoning and crowdsourcing techniques for large-scale entity linking. In: WWW, pp. 469–478 (2012)
7. Fang, Y., Si, L., Yu, Z., et al.: Purdue at TREC 2010 Entity Track: A Probabilistic Framework for Matching Types Between Candidate and Target Entities. In: TREC (2010)
8. Finkel, J.R., Grenager, T., Manning, C.: Incorporating non-local information into information extraction systems by gibbs sampling. In: ACL, pp. 363–370 (2005)
9. Finkel, J.R., Manning, C.D.: Joint parsing and named entity recognition. In: NAACL, pp. 326–334 (2009)
10. Gangemi, A., Nuzzolese, A.G., Presutti, V., Draicchio, F., Musetti, A., Ciancarini, P.: Automatic typing of dbpedia entities. In: Cudré-Mauroux, P., Heflin, J., Sirin, E., Tudorache, T., Euzenat, J., Hauswirth, M., Parreira, J.X., Hendler, J., Schreiber, G., Bernstein, A., Blomqvist, E. (eds.) ISWC 2012, Part I. LNCS, vol. 7649, pp. 65–81. Springer, Heidelberg (2012)
11. Holmes, G., Hall, M., Frank, E.: Generating rule sets from model trees. In: Foo, N.Y. (ed.) AI 1999. LNCS, vol. 1747, pp. 1–12. Springer, Heidelberg (1999)
12. Järvelin, K., Kekäläinen, J.: Cumulated gain-based evaluation of IR techniques. In: TOIS, pp. 422–446 (2002)
13. Kalyanpur, A., Murdock, J.W., Fan, J., Welty, C.: Leveraging community-built knowledge for type coercion in question answering. In: Aroyo, L., Welty, C., Alani, H., Taylor, J., Bernstein, A., Kagal, L., Noy, N., Blomqvist, E. (eds.) ISWC 2011, Part II. LNCS, vol. 7032, pp. 144–156. Springer, Heidelberg (2011)
14. Kumar, R., Tomkins, A.: A characterization of online search behavior. IEEE Data Eng. Bull. (2009)
15. Liu, T.-Y.: Learning to rank for information retrieval. In: FTIR, pp. 225–331 (2009)
16. Matuszek, C., Cabral, J., Witbrock, M., DeOliveira, J.: An introduction to the syntax and content of cyc. In: AAAI Spring Symposium (2006)
17. Mühleisen, H., Bizer, C.: Web data commons - extracting structured data from two large web corpora. In: LDOW (2012)
18. Nadeau, D.: Semi-supervised named entity recognition: learning to recognize 100 entity types with little supervision. PhD thesis (2007)
19. Nadeau, D., Turney, P.D., Matwin, S.: Unsupervised named-entity recognition: generating gazetteers and resolving ambiguity. In: Lamontagne, L., Marchand, M. (eds.) Canadian AI 2006. LNCS (LNAI), vol. 4013, pp. 266–277. Springer, Heidelberg (2006)
20. Pound, J., Mika, P., Zaragoza, H.: Ad-hoc object retrieval in the web of data. In: WWW, pp. 771–780 (2010)
21. Quinlan, J.R.: Learning with continuous classes. In: AI, pp. 343–348 (1992)
22. Suchanek, F.M., Abiteboul, S., Senellart, P.: Paris: Probabilistic alignment of relations, instances, and schema. In: PVLDB, pp. 157–168 (2011)

23. Suchanek, F.M., Kasneci, G., Weikum, G.: Yago: a core of semantic knowledge. In: WWW, pp. 697–706 (2007)
24. Tonon, A., Demartini, G., Cudré-Mauroux, P.: Combining inverted indices and structured search for ad-hoc object retrieval. In: SIGIR, pp. 125–134 (2012)
25. Tylenda, T., Sozio, M., Weikum, G.: Einstein: physicist or vegetarian? summarizing semantic type graphs for knowledge discovery. In: WWW, pp. 273–276 (2011)
26. Welty, C., Murdock, J.W., Kalyanpur, A., Fan, J.: A comparison of hard filters and soft evidence for answer typing in watson. In: Cudré-Mauroux, P., et al. (eds.) ISWC 2012, Part II. LNCS, vol. 7650, pp. 243–256. Springer, Heidelberg (2012)
27. Whang, S.E., Garcia-Molina, H.: Joint Entity Resolution on Multiple Datasets. The VLDB Journal (2013)

DynamiTE: Parallel Materialization of Dynamic RDF Data

Jacopo Urbani, Alessandro Margara, Ceriel Jacobs,
Frank van Harmelen, and Henri Bal

Vrije Universiteit Amsterdam, The Netherlands
{jacopo,ceriel,frank.van.harmelen,bal}@cs.vu.nl, a.margara@vu.nl

Abstract. One of the main advantages of using semantically annotated data is that machines can *reason* on it, deriving implicit knowledge from explicit information. In this context, *materializing* every possible implicit derivation from a given input can be computationally expensive, especially when considering large data volumes.

Most of the solutions that address this problem rely on the assumption that the information is static, i.e., that it does not change, or changes very infrequently. However, the Web is extremely dynamic: online newspapers, blogs, social networks, etc., are frequently changed so that outdated information is removed and replaced with fresh data. This demands for a materialization that is not only *scalable*, but also *reactive* to changes.

In this paper, we consider the problem of *incremental materialization*, that is, how to update the materialized derivations when new data is added or removed. To this purpose, we consider the ρdf RDFS fragment [12], and present a parallel system that implements a number of algorithms to quickly recalculate the derivation. In case new data is added, our system uses a parallel version of the well-known semi-naive evaluation of Datalog. In case of removals, we have implemented two algorithms, one based on previous theoretical work, and another one that is more efficient since it does not require a complete scan of the input.

We have evaluated the performance using a prototype system called *DynamiTE*, which organizes the knowledge bases with a number of indices to facilitate the query process and exploits parallelism to improve the performance. The results show that our methods are indeed capable to recalculate the derivation in a short time, opening the door to reasoning on much more dynamic data than is currently possible.

1 Introduction

One of the main advantages of using semantically annotated data is that machines can *reason* on it, deriving new, implicit, knowledge from existing information. To this end, several systems have been developed over the last few years to make all possible conclusions from a given input explicit, so that no reasoning is needed when the user queries the knowledge base.

This task, also called *materialization*, can be computationally expensive, especially if the input is large. In fact, while some of these systems can perform

H. Alani et al. (Eds.): ISWC 2013, Part I, LNCS 8218, pp. 657–672, 2013.

the materialization of several billions of statements [10,16,20], they still demand for significant computational resources, and the process may require up to a few days to complete. Because of this, most of these systems work under the assumption that the input data is static, i.e., that it does not change, or changes very infrequently. This assumption does not match with the current Web, which is extremely dynamic: online newspapers, blogs, social networks, are frequently changed and updated. This demands for a materialization process that is not only scalable, but also *reactive* to changes, by reducing the cost of updating the materialization. A new research area, called *stream reasoning*, has recently emerged to address this specific problem [4].

With this paper, we contribute to this area by considering the problem of *incrementally maintaining* a large materialized knowledge base in the presence of frequent changes, using monotonic rule-based reasoning as the method to derive new information. More specifically, we propose *DynamiTE*, a parallel system capable of efficiently generating the complete materialization of large RDF knowledge bases, and maintaining it after the knowledge base is updated.

We consider two types of updates: (*i*) the *addition* of new information, which requires a re-computation of the materialization to add new derivations, and (*ii*) the *removal* of existing information, which requires the deletion of the explicit knowledge, and also of all the implicit information that is no longer valid.

For the addition updates, *DynamiTE* applies in parallel the well-known Datalog semi-naive evaluation. For the removal updates, it implements two algorithms: one that was presented in the literature but only from a theoretical perspective [11], and another one, which is more efficient since it does not require a complete scan of the input for every update.

To evaluate the performance of *DynamiTE*, we consider the minimal RDFS fragment ρdf [12], which captures the main semantic functionality of RDFS [9] limiting the materialization accordingly. Furthermore, its ruleset can be expressed with Datalog [1] and this allows us to reuse its theory to define the semantics of our process. We designed some experiments to study the behavior of our system over large amounts of data, trying to understand, from a system point of view, what are the main factors that drive the performance. The results show that our system is capable of efficiently computing the materialization of large knowledge bases up to one billion statements, and can alter them in a range from hundred of milliseconds to less than two minutes when considering substantial updates of two hundred thousand triples.

The remaining of this paper is organized as follows: Section 2 contains some background information to make the reader familiar with the concepts we use throughout the paper. Next, Section 3 reports an overview of our system while Sections 4 and 5 focus on the crucial task of the system, i.e., the incremental maintenance of the materialization. After this, we present an evaluation of our system in Section 6. Finally, Section 7 discusses related work and Section 8 concludes the paper, also reporting some indications for future research.

2 Background

To describe our system, we use the notions and notations of the Datalog language. Because of space constraints, we cannot present a complete overview of this language. Therefore, we only present some basic concepts that we use in the paper, and we refer the reader to existing literature, e.g. [1], for more details.

First of all, a Datalog *program* P is defined as a finite set of rules in the form $R_1(w_1) \leftarrow R_2(w_2), R_3(w_3), ..., R_n(w_n)$, where each component $R_i(w_i)$ is called a *literal*. A literal is composed of a *predicate* (e.g. R_i) and a tuple of terms $w_i := t_1, ..., t_m$. A term t_i can be either a variable from a finite domain V or a constant term from another disjoint finite domain C. We denote with $t_j \in R_i(w_i)$ the term that appears at the j^{th} position of w_i.

We define $var(R_i(w_i))$ as the set of all variables in the literal $R_i(w_i)$, and $const(R_i(w_i))$ as the set of all its constants. The left side of a rule r is called the *head* of r ($head(r)$), while the right side is defined as its *body* ($body(r)$). Datalog imposes that each variable that appears in the head of the rule must also appear in its body. This means that $\forall v \in var(R_1(w_1))$ there must be an $i \in (2..n)$ so that $v \in var(R_i(w_i))$. Furthermore, Datalog makes a distinction between *edb* predicates, which never appear in the head of a rule, and *idb* predicates, which appear in the head of some rule.

A literal containing only constants is called a *fact*. We say that a fact f *instantiates* the literal l if they share the same predicate, every $c_i \in const(l)$ that appears in l at position i is equal to the corresponding term t_i of f, and if there is a variable $v \in var(l)$ which appears in l at two different positions i and j, then $t_i = t_j$ in f. We call $f_1 \leftarrow f_2, f_3, ..., f_n$ an *instantiation* of a rule $R_1(w_1) \leftarrow R_2(w_2), R_3(w_3), ..., R_n(w_n)$ if every $f_{i \in \{1..n\}}$ is an instantiation of the corresponding $R_i(w_i)$, and that any term $t_j \in f_i$ is equal to another term $t_m \in f_{j \in \{1..n\} \wedge i \neq j}$ if $v_i \in var(R_i(w_i))$, $v_j \in var(R_j(w_j))$, and $v_i = v_j$.

In Datalog, the operator T_P is called the *immediate consequence operator* of P. This operator maps a generic database $\mathfrak{I}$ (defined as a finite set of facts) to another database $T_P(\mathfrak{I})$ that contains all facts that are *direct consequences* for $\mathfrak{I}$ and P. A fact f is a direct consequence for $\mathfrak{I}$ and P if either $f \in \mathfrak{I}(R)$[1] for some *edb* predicate R in P or $f \leftarrow f_1, f_2, ..., f_n$ is an instantiation of a rule in P and each $f_{i \in \{1..n\}} \in \mathfrak{I}$. Intuitively, T_P can be seen as the abstract operator that applies the rules in P over $\mathfrak{I}$ to derive new conclusions.

T_P is monotonic: given two databases $\mathfrak{I}, \mathfrak{I}'$, if $\mathfrak{I} \subseteq \mathfrak{I}'$, then $T_P(\mathfrak{I}) \subseteq T_P(\mathfrak{I}')$. Because of this, repeated executions of the T_P operator over augmented versions of a database $\mathfrak{I}$ will lead to a fix-point. This means that if we define $T_P^n(\mathfrak{I})$ as

$$T_P^n(\mathfrak{I}) = \begin{cases} \mathfrak{I} & n = 0 \\ T_P(T_P^{n-1}(\mathfrak{I})) & n > 0 \end{cases}$$

there is a n' such that $T_P^{n'+1}(\mathfrak{I}) = T_P^{n'}(\mathfrak{I})$. In this case, $T_P^{n'}(\mathfrak{I})$ is named as the fix-point of T_P, and denoted with $T_P^{\omega}(\mathfrak{I})$. $T_P^{\omega}(\mathfrak{I})$ is the *materialization* of $\mathfrak{I}$, since it contains all the possible derivations that can be obtained from $\mathfrak{I}$ and P.

[1] The database $\mathfrak{I}(R)$ contains all and only the facts $f \in \mathfrak{I}$ with predicate R.

Table 1. Supported ruleset. Datalog variables are in italics; constants are in fixed-width characters. The abbreviations `SPO, SCO, TYPE, RANGE` and `DOMAIN` stand for the URIs *rdfs:subProperty*, *rdfs:subClassOf*, *rdf:type*, *rdfs:range*, and *rdfs:domain*. The first rule is only used to map the *edb* predicate T_e to the *idb* predicate T_i that is used in the other rules.

Head	**Body**
$T_i(A, P, B)$	$\Leftarrow T_e(A, P, B)$
$T_i(A, \texttt{SPO}, C)$	$\Leftarrow T_i(A, \texttt{SPO}, B), T_i(B, \texttt{SPO}, C)$
$T_i(A, P, B)$	$\Leftarrow T_i(Q, \texttt{SPO}, P), T_i(A, Q, B)$
$T_i(A, \texttt{TYPE}, C)$	$\Leftarrow T_i(B, \texttt{SCO}, C), T_i(A, \texttt{TYPE}, B)$
$T_i(A, \texttt{SCO}, C)$	$\Leftarrow T_i(A, \texttt{SCO}, B), T_i(B, \texttt{SCO}, C)$
$T_i(A, \texttt{TYPE}, D)$	$\Leftarrow T_i(P, \texttt{DOMAIN}, D), T_i(A, P, B)$
$T_i(A, \texttt{TYPE}, R)$	$\Leftarrow T_i(P, \texttt{RANGE}, R), T_i(B, P, A)$

A trivial way to compute $T_P^\omega(\mathfrak{I})$ is to start from $n = 0$, execute T_P, and increase n until the fix-point is reached. This approach, known as the *naive* evaluation, is very inefficient since, at each iteration, an application of T_P will recompute all the derivations already computed in the previous iterations.

A more efficient algorithm, called *semi-naive* evaluation [1], optimizes this process by instantiating a rule r only if at least one fact that instantiates a literal in $body(r)$ was derived in the previous iteration. In this way, the algorithm is able to significantly decrease the number of duplicates.

The semi-naive evaluation can be implemented by annotating each fact in the database with a numeric step that indicates at which stage of the derivation that information was derived (facts in the original input are marked with a step of zero). Then, at every n^{th} iteration of the evaluation, the operator T_p accepts only instantiations if at least one fact that instantiates a literal has a step that is equal or greater than $n - 1$. This significantly reduces the number of rules execution and consequently the number of duplicates that are generated.

3 DynamiTE: System Overview

The purpose of *DynamiTE* is to efficiently compute and incrementally maintain the materialization of a database, which consists of RDF triples. We consider the minimal RDFS fragment ρdf [12], and execute the rules presented in Table 1.

To formalize our problem in Datalog, let P be a program consisting of the rules in Table 1, and $\mathfrak{I}$ a given database, which represents the initial RDF knowledge base expressed as a set of Datalog facts $T_e(s, p, o)$ where T_e is an *edb* predicate, and s, p, o are respectively the subject, predicate, and object of a RDF triple that are mapped to constant terms in Datalog.

DynamiTE implements three main tasks: (*i*) First, it computes the complete materialization of $\mathfrak{I}$. Then, it maintains it after a set of triples δ is either (*ii*) added, or (*iii*) removed. More formally, in (*i*) the system calculates $T_P^\omega(\mathfrak{I})$, in (*ii*) $T_P^\omega(\mathfrak{I} \cup \delta)$, and in (*iii*) $T_P^\omega(\mathfrak{I} \setminus \delta)$ is computed.

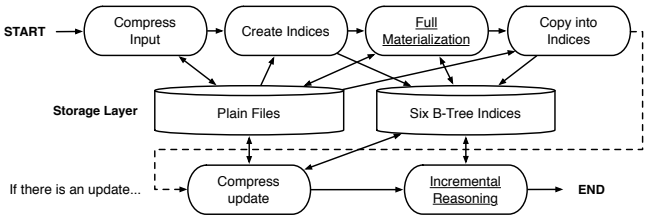


Fig. 1. *DynamiTE*: General System Workflow

3.1 System Workflow

Fig. 1 shows the general workflow of *DynamiTE*. The initial input consists of a collection of triples encoded with the N-Triples format. First, *DynamiTE* performs the *Compress Input* operation, which converts the textual terms into numbers using the technique of dictionary encoding. The algorithm used for this phase is an adaptation of the distributed MapReduce version presented in [17].

The next operation, *Create Indices*, stores the compressed data into B-Tree indices (along with the dictionary tables to allow quick decompression). Since the size of the database can easily become too large to fit in main memory, we need to consider data structures that can be off-loaded to disk. To this end, we use six on-disk B-Trees to store all the possible permutations of the input triples. We chose the B-Tree data structure because we want our system to support generic querying, and storing six indices has proven to be ideal for SPARQL [14] querying, allowing an efficient retrieval for all possible atomic queries [21]. As we will show in the evaluation, the *Create Indices* operation is the most expensive of the entire workflow. To reduce its cost, *DynamiTE* sorts the triples before insertion (in our tests, this improves the performance by at least 10%).

Next, *DynamiTE* performs the *Full Materialization*. We describe this process in detail in Section 4.1. This operation implements a parallel version of the semi-naive evaluation, which iteratively reads the entire input and augments it with new derivations. The B-Trees are not particularly efficient in supporting these operations. Therefore, during the full materialization, the system writes the new derivation on plain files, and copies them on the B-Trees in the next operation, *Copy into Indices*. According to some tests we performed, using files makes the entire process at least 30% faster.

At this point, *DynamiTE* is ready to receive updates (see Fig. 1, bottom). For each update δ, it first compresses the content of δ, and then performs incremental reasoning. We describe this last operation in Sections 4 and 5.

3.2 Physical Rules Instantiation

In order to physically instantiate the rules, we make a fundamental distinction between *schema* and *generic* triples. We denote as *schema* all those triples having `SPO`, `SCO`, `DOM`, or `RANGE` as predicate. We call all the others *generic* triples. The design and implementation of our algorithm for rules instantiation relies on the

assumption that the number of schema triples in the input is significantly smaller than the rest, so that all of them (explicit and inferred) will fit in main memory. This assumption holds for the vast majority of web data [16], but there can be scenarios where this is no longer true.

First, the system assigns all the rules to three disjoint subsets, named *Type 1*, *Type 2*, and *Type 3*, depending on the number and type of their literals. The assignment criteria for a rule r are defined as follows:

- *Type 1.* All the literals in r can be instantiated only by schema triples.
- *Type 2.* The body of r consists of only one literal and can be instantiated by generic triples.
- *Type 3.* The body of r contains exactly two literals, one of them can be instantiated only by schema triples while the other is instantiated by generic triples.

DynamiTE implements a different rules instantiation strategy depending on the type of the rules. Rules of *Type 1* are instantiated by first loading all schema triples, i.e., all the triples that can instantiate the literals in $body(r)$, in memory. Then, in case there is only one literal in $body(r)$ the instantiation becomes trivial since the system only needs to generate a new triple that instantiates $head(r)$ and copy the values of the variable in the body to the corresponding position in the head. If there are multiple literals in the body, then the system must join the triples having common terms. Consider for example the second rule in Table 1: its application needs to find all the pairs of triples s, p, o, and s', p', o', such that $p = p' = \texttt{SPO}$ and $o = s'$. The system performs this operation in memory, computing a hash join between the two sets of triples.

Rules of *Type 2* are similar to rules of *Type 1* with only one literal in their body. The only difference is that here the system cannot assume that they fit into the main memory and thus it needs to retrieve them from the disk.

Rules of *Type 3* are the most challenging, since they require a join between two sets of triples, schema and generic triples, where the second set can be quite large. In previous work, we tackled this problem proposing a distributed execution over multiple processing nodes using the MapReduce model and the Hadoop framework [16]. Schema triples were replicated on every node, and a hash join was performed against the generic triples that were being read from the input. While this approach proved to be scalable, it introduced high latency (due to the usage of Hadoop), which conflicts with our need for reactivity. In *DynamiTE*, we re-implemented similar algorithms to replicate the MapReduce programming model without using a resilient and distributed architecture such as Hadoop, but instead exploiting the parallelism offered by modern multi-core hardware to reduce the processing time.

4 Materialization after Data Additions

We distinguish two types of updates, depending on whether the initial database is empty. In the first case we perform a full materialization, while in the second an incremental materialization is done.

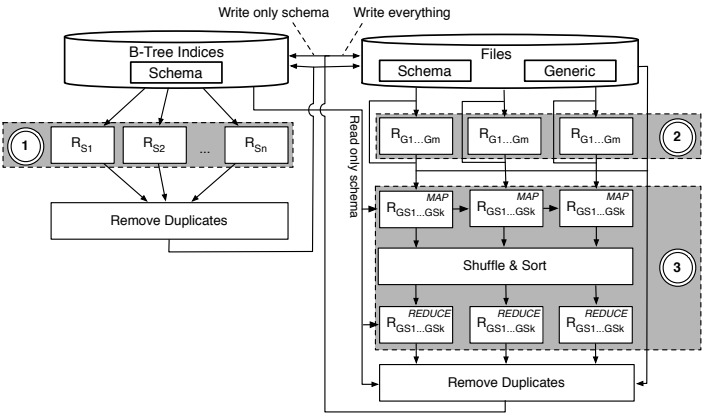


Fig. 2. *DynamiTE*: One iteration of the full materialization. In this figure, there are n rules of *Type 1*, $R_{S1}...R_{Sn}$, m rules of *Type 2*, $R_{G1}...R_{Gm}$, and k rules of *Type 3*, $R_{GS1}...R_{GSk}$.

4.1 Full Materialization

For this task, *DynamiTE* provides a parallel implementation of the semi-naive evaluation to exploit the parallelism of modern computer architectures. As we described in Section 2, the semi-naive evaluation performs multiple iterations, until it reaches a fix-point. Fig. 2 shows a single iteration in our system. On top of the figure there is the database, physically stored both on six B-Trees and on a number of files. During the full materialization, we read the database and write the derivation only to files, except for the schema triples that are always being replicated on the B-Trees to improve their retrieval in the next iterations.

First, *DynamiTE* applies all the rules of *Type 1*. This step is shown in the gray box marked with a "1". The execution is parallel, with each rule r being instantiated in a separate thread t. Each thread t retrieves from the B-Trees the schema triples that instantiate the literals in $body(r)$, joins them, if needed, and generates the triples that instantiate $head(r)$. Finally, it stores all derivations both in the B-Trees and on files.

Next, *DynamiTE* instantiates rules of *Type 2* and *Type 3*. They require a complete scan over the input, which can potentially be large. *DynamiTE* optimizes this step by partitioning the input files into smaller blocks, with each block b being read by a different thread t. First, each thread t applies the rules of *Type 2* on the triples in b. Then, it applies rules of *Type 3*, considering both the original input and the output of the rules of *Type 2*. Rules of *Type 3* are instantiated using the MapReduce algorithm outlined in Fig. 2. Notice that, for executing rules of *Type 3*, *DynamiTE* also accesses the B-Trees for retrieving schema triples. After execution, all the derivations are stored on files, while schema triples are also replicated in the B-Trees.

As mentioned, *DynamiTE* implements the semi-naive evaluation, where only the last derivations are considered during rule execution. To make this possible,

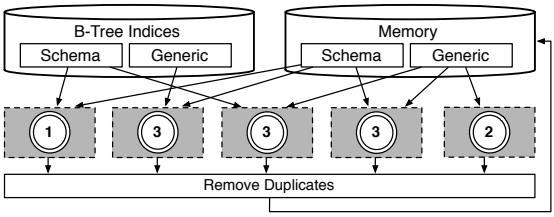


Fig. 3. *DynamiTE*: One iteration of the incremental materialization

it marks each triple with a *step* attribute, representing the step when it was first derived. For example, at the first iteration all the triples derived by rules of *Type 1* have step one; the derivations from rules of *Type 2* are marked with step two, and so on. After the first iteration, the system accepts a derivation only if at least one of the triples that instantiate a literal in the body of the rule has a step greater or equal than the current one minus three.

The algorithm stops iterating the rules instantiation when none of them derived new triples. As described in the previous section (see Fig. 1), after the materialization is complete, all the derived triples are copied into the B-Tree indices for efficient querying.

4.2 Incremental Materialization

The previous section showed how *DynamiTE* computes the full materialization of a knowledge base and stores it into B-Tree indices. Now, we show how it maintains this materialization in the presence of data additions.

Our system performs this operation incrementally, fully exploiting the existing materialization $T_P^\omega(\mathfrak{I})$. The process can be divided into three main steps: (*i*) Load the update into a set called δ, which is stored in main memory. (*ii*) Perform a semi-naive evaluation on $T_P^\omega(\mathfrak{I}) \cup \delta$ to derive new triples. In this case a rule is instantiated only if at least one fact is contained in δ. (*iii*) Add all the new derivations into the B-Tree indices, making them available for querying.

Fig. 3 shows how the semi-naive evaluation in phase (*ii*) is implemented in *DynamiTE*. Every gray block in Fig. 3 is implemented as shown in Fig. 2 and is executed in parallel. At every iteration, these blocks consider only rules where at least one literal in the body can be instantiated from a triple in δ, and might produce new derivations that become the new δ in the next iteration. Moreover, all previously derived triples remain available in main memory since they might contribute in the following iterations to produce new derivation.

The first block considers rules of *Type 1* and reads schema triples from both memory and B-Trees. The last block reads only the generic triples in δ and executes rules of *Type 2* on them. We split the execution of rules of *Type 3* into three blocks: one reads both schema and generic triples from memory; one reads only schema triples in δ and the generic triples from the B-Trees; the last one reads schema triples from the B-Trees and generic triples from δ. This division is important since it allows us to significantly reduce the amount of

information read from the B-Trees, and hence from $T_P^\omega(\mathfrak{I})$, to only the triples that can produce new derivations. This is achieved by first reading a triple t from δ, and then retrieving from the B-Trees only the triples that can be joined with t to complete the instantiation.

After launching the execution of these blocks on different threads, *DynamiTE* waits for them to finish, removes the duplicates, and continues to iterate until no new derivation is produced.

5 Materialization after Data Removals

When removing a set of triples δ from a database $\mathfrak{I}$, we also need to remove from the materialized view all the triples that cannot be derived from $\mathfrak{I} \setminus \delta$. In this section, we describe the implementation of two algorithms, one already described in [11] (but only from a theoretical perspective) and another one that is based on the idea of counting all the direct derivations.

5.1 DRed Algorithm (and Derivatives)

The first algorithm implemented by *DynamiTE* is known as *Delete and Rederive* (*DRed*) and was first introduced in [7]. It works in two steps: (*i*) First, it computes all the triples that can be derived from δ, and removes them from the knowledge base. This process clearly computes an overestimation of the triples to remove, since some of them can have alternative derivations in $\mathfrak{I} \setminus \delta$. Therefore, as a second step, (*ii*) the algorithm re-derives the triples that are still valid, and adds them again to the knowledge base.

Our implementation is based on a similar version of DRed, presented in [11], which has the advantage of using only the original set of rules for maintaining the materialization. More precisely, it implements the first phase (*Delete*) as a semi-naive evaluation that only considers rule evaluations in which at least one literal in the body of the rule is instantiated from a triple in δ (or derived from δ in previous iterations). All the triples derived in this phase are removed from the knowledge base. The second phase (*Rederive*) is again implemented as a semi-naive evaluation, which considers all the triples left in the knowledge base.

Since we assume that the size of our update (and the derivations it produces) is small, we first load δ in memory. Then, we start executing the *Delete* phase: we do so by using the implementation of the semi-naive evaluation presented in Fig. 3. This means that we consider only derivations that involve triples in δ (or derived from δ in previous steps) and we store the output of the computation in memory, until we reach the fix-point.

After the first phase completes, we remove all the derivations stored in memory from the knowledge base, and we start the second, *Rederive* phase. Here too, we exploit the implementation of the semi-naive evaluation described above. This step outputs (and stores in main memory) all the triples that were removed from the knowledge base in the *Delete* phase, but that could actually be derived from $\mathfrak{I} \setminus \delta$. As a final step, we add them again into our knowledge base.

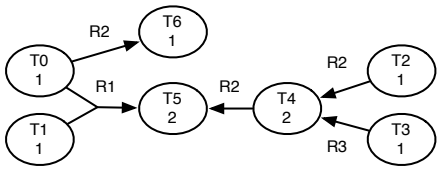


Fig. 4. Counting algorithm. An example of how the count attribute is computed. Nodes represent triples and include their name and their count. Arrows represent applications of rules and are labeled with the name of the rule.

5.2 Counting Algorithm

During the *Delete* phase, the previous algorithm computes all the triples that can be derived using triples in δ. Let us call this set T_δ. We call $D_\delta \subseteq T_\delta$ the subset of triples that cannot be derived from $\mathfrak{I} \setminus \delta$, and $A_\delta = T_\delta \setminus D_\delta$. While only the triples in D_δ have to be removed from the knowledge base, the previous algorithm does not have enough information for recognizing them. For this reason, it removes all the triples in T_δ and then recalculates the set A_δ during the *Rederive* phase.

To improve the performance, we propose an alternative method, where all the triples are annotated with additional information that allows for immediate discrimination between the two sets A_δ and D_δ. This additional information consists of a new *count* attribute, which represents the number of possible rule instantiations that produced t as a *direct consequence*, plus one if the triple was also present in the original input.

As an example of how the count attribute is computed, consider Fig. 4. The figure shows a simple graph of derivations, where nodes represent triples and arrows represent applications of rules. For instance, `T5` can be derived by applying rule `R1` to `T0` and `T1`. `T0`, `T1`, `T2`, and `T3` are the facts in the original input. They are stored in the knowledge base with count equal to one. `T4` can be derived from rule `R2` using `T2`, and from rule `R3` using `T3`, so its count is two. Similarly, `T5` can be derived using `R1` from `T0` and `T1`, and from `T4` with `R2`, so its count is two. Finally, `T6` can be derived from `T0` and has count equal to one. Notice that we consider only *direct* derivations: although `T4` can be derived in two ways, it participates in the count of `T5` only once.

During the *Delete* phase, the presence of the count attribute enables us to discriminate between triples in D_δ and A_δ. As an example, consider again the graph in Fig. 4, and assume that we want to remove `T0`, i.e., $\delta = \{\texttt{T0}\}$.

We start a semi-naive evaluation to compute all the triples that can be derived from `T0`, i.e., `T5` and `T6`, and we decrement their counts by one. All the triples whose count goes to zero (`T6`, in our example) do not have alternative derivations. They belong to D_δ, and can be removed from the knowledge base. On the other hand, `T5` has count two in the materialized database: by removing `T0`, its count is decreased by one, but still there is one derivation left. This means that `T5` is part of A_δ (i.e., it can still be derived from $\mathfrak{I} \setminus \delta$), and should not be removed from the materialized database.

In this simple example, the semi-naive evaluation just required one iteration to reach the fix-point. If more iterations are needed the algorithm works as follows: at iteration n, it considers only rule instantiations that involve triples that were actually removed from the knowledge base at iteration $n-1$, i.e., triples whose count went to zero at iteration $n-1$.

Using this algorithm, the *Delete* phase computes *only* the triples that actually need to be removed from the knowledge base, i.e., the triples in D_δ. As a consequence, we can skip the *Rederive* phase.

Finally, notice that the counting algorithm requires the *count* attribute to be computed and maintained for each triple in the knowledge base. To do so, we implemented a slightly different version of the algorithms described in the previous sections for computing the (complete or incremental) materialization in case of data additions. In particular, after each derivation step, we never remove duplicates. Instead, every time we add a triple to the knowledge base, we check whether it was already present or not. If it is new, we add it with a count of one; otherwise, we increase its count by one.

6 Evaluation

Our evaluation has two goals: first, we want to test the absolute performance of *DynamiTE* when considering the computation of a full materialization and its maintenance in case of updates. Second, we want to compare the behavior of an existing state of the art algorithm for incremental update, namely DRed, with the other counting algorithm. To perform the experiments, we used one machine in the DAS-4 cluster[2], which is equipped with a dual quad-core Intel E5620 CPU, 24GB of main memory, two hard disks of 1TB connected with RAID-0, and one 500GB SSD disk.[3].

Dynamite is fully written in Java[4], and it uses BerkeleyDB Java Edition [13] as implementation of the on-disk B-Trees. We chose BerkeleyDB since it is among the most widely used databases, and it fully supports many functionalities such as transactions and concurrency. We use the LUBM [5] benchmark test to evaluate the performance of our system. We chose this dataset for two reasons: (*i*) it is one of the de-facto standard benchmarks to test the performance of reasoning on RDF data; (*ii*) it allows us to tune the experiments to control the amount of derivation that is produced.

We present three sets of experiments, with the following goals: (*i*) to evaluate the costs of performing the complete materialization; (*ii*) to evaluate the cost of the updates, and (*iii*) to discover the performance bottlenecks of our system.

Complete Materialization. Fig. 5 (left) shows the execution time required by each single task of the initial workflow (see Fig. 1) to materialize the LUBM(1000) dataset, which contains about 138 million triples. From this table, we notice that

[2] `http://www.cs.vu.nl/das4`
[3] In some experiments the machine had only a 256GB SSD disk.
[4] The code is available online at `http://github.com/jrbn/dynamite`

Tasks	Time (sec.)
Compress Input	876
Create Indices	7953
Full Materializ.	886
Copy to Indices	5967

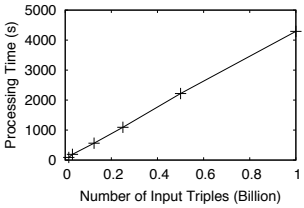


Fig. 5. (Left) Execution time for all the tasks during the loading of the database. (Right) Execution time of the complete closure.

both compression and materialization take a relatively short time compared to the operation of copying the triples into the B-Tree indices. This last task is not related to reasoning, yet it is a necessary part in our workflow.

To better understand the performance of our system in computing the full materialization, we study this single operation in more detail. To this purpose, Fig. 5 (right) shows how the execution time for the full materialization changes with the number of triples present in the original input, starting from LUBM(125), consisting of 16 million triples, to LUBM(8000), of about one billion triples. We observe that our implementation has good scalability, since the execution time increases linearly with respect to the size of the input. Furthermore, the system has a high throughput: it computes the closure of about one billion LUBM triples in about 4400 seconds, which results in an input processing ratio of about 227K triples/sec. As an informal comparison, the throughput per machine of WebPIE [16] to materialize the same dataset with RDFS was about 55K triples/sec, four times lower than *DynamiTE*.

Incremental Updates. After the system has materialized the input knowledge base, the user can update it by removing or adding new triples. Unfortunately, to the best of our knowledge there is no benchmark tool to evaluate reasoning on a sequence of additions and removals. Because of this, to evaluate the performance of this operation, we created six different type of updates from the LUBM dataset:[5]

- **Update 1.** Add/Remove one triple, which does not trigger any reasoning.
- **Update 2.** Add/Remove $\sim$ 16k triples, which do not trigger any reasoning.
- **Update 3.** Add/Remove $\sim$ 8k triples with the predicate `LUBM:emailAddress`. This update triggers reasoning so that a fixed number of triples is derived.
- **Update 4.** Remove the triple indicating that the property `LUBM:headOf` is a subproperty of `LUBM:worksFor`. This removal triggers a reasoning process that derives a number of triples proportional to the input size. To simulate the same reasoning for the addition, we add two new triples that indicate that `LUBM:headOf` is a subproperty of a new property `LUBM:responsibleOf`, which is also a subproperty of a new `LUBM:Manager`.
- **Update 5 and 6**. Add/Remove respectively one and two entire universities.

[5] For reproducibility, also these updates are available in the repository of the project.

We chose Update 1 and Update 2 to evaluate the insertion cost and the overhead of checking that no reasoning can be applied. Update 3 represents a small update that produces only limited reasoning. Update 4 consists of schema information, which has a consistent impact on the knowledge base. Update 5 and Update 6 represent large updates, which trigger the execution of multiple rules. Notice that significantly larger updates are not possible since we store the update in main memory. Such updates can be handled either by splitting them in chunks so that the produced data can fit in memory, or by re-launching a full materialization which can be more efficient since it reads the input from plain files rather than the B-Trees.

Table 2 shows the execution time of these updates using the three algorithms explained above. We noticed a certain fluctuation in the runtime, so we repeated every measurement five times and report their average. First of all, we notice that the runtime for the addition ranges from 117 ms (insertion of one triple) to 31.8 s (addition of two universities, i.e., $\sim$ 250k triples). Even though these runtimes cannot always guarantee a real-time processing, they are significantly lower than recomputing a complete materialization. A significant result is represented by the results obtained in Update 1 and in Update 2. In both cases, no reasoning is triggered; however, Update 2 is significantly slower, since it requires *DynamiTE* to add 16k triples into the BTrees. Once again, this experiment demonstrates how accessing the B-Trees on disk represents the main cost for *DynamiTE*.

Considering the removal, we immediately observe that the DRed algorithm is very slow, with a runtime that is always larger than 30 minutes. This is due to the fact that the *Rederive* phase needs to access the entire input to recompute possible conclusions that were incorrectly removed in the *Delete* phase. In contrast, the counting algorithm is much faster, with a runtime that ranges from 117 ms (removal of a triple) to 135 s (removal of two universities). We must remark that the procedure of enabling the counting slows down the initial preprocessing by about 49%[6]. However, the advantage obtained during a removal is so significant that this additional cost is quickly amortized after only a few updates.

Performance Bottleneck. To further investigate the behavior of *DynamiTE*, and to identify the main performance bottlenecks, we launched the complete materialization and the incremental Update 4 changing two critical settings of the system: the kind of disk adopted and the number of threads used for processing.

First, we changed the disk storage from an SSD to a normal HDD. The full closure was only 6% slower, but the incremental addition and removal became 30 times and 15 times slower, respectively. This clearly indicates that the disk speed is a performance bottleneck for accessing and updating the B-Tree indices.

Second, we decreased the degree of parallelism by decreasing the number of concurrent processing threads from eight to four (the SSD disk was used for the storage). The runtime of the materialization became 12% slower, while the runtime of the incremental addition and removal became respectively at most 6% and 30% slower.

[6] This overhead is already included in the results presented in Fig. 5.

Table 2. Runtime of four type of updates on LUBM(1000) after a complete materialization that required about 15 minutes (see Fig. 5)

Update	Addition (sec.)	Removal (sec.)	
		DRed	Counting
Update 1	0.117	2902.7	0.117
Update 2	8.2	2049.6	25.7
Update 3	3.7	2121.9	25.4
Update 4	31.8	2132.2	51.0
Update 5	16.8	2196.0	74.6
Update 6	30.5	3830.2	135.8

From these experiments, we can conclude that the degree of parallelism is an important component in shaping the performance, especially for the full materialization. However, disk throughput remains the largest performance bottleneck for the incremental updates, since the disk-based data structure heavily relies on it to retrieve the data.

7 Related Work

The problem of updating derived information upon changes in the knowledge base has been widely studied by both the AI and database communities in the contexts of truth maintenance [3], view maintenance [6], and deductive databases [15]. In both areas, the idea of *incrementally* updating derived information has been studied since the beginning of the 1980s, and in the database community it led to two main algorithms: DRed (Delete and Rederive) [7], and PF (Propagate Filter) [8]. Both algorithms share the same idea: when some base facts are removed, they first compute an overestimation of the derived knowledge that needs to be deleted, and then rederive the information that is still valid.

A declarative version of the DRed algorithm was first introduced in [15] and then extended in [18,19] to consider also updates in the ruleset. These algorithms, however, create an update program that can be significantly larger than the original program (in terms of number of rules). Moreover, the update program can include negations, even if they are not present in the original program. Our implementation of the DRed algorithm follows the work presented in [11], which overcomes the limitations listed above and manages incremental updates without changing the set of derivation rules. As we show in the evaluation, DRed has the disadvantage of always requiring to read full knowledge base.

The counting algorithm is significantly more efficient than DRed, since it avoids (when possible) a complete scan over the input. This algorithm is based on the operation of counting and decrementing the number of possible derivations. This idea was also introduced in the original DRed paper [7], but it was not implemented nor designed for a declarative language.

Recently, new solutions for incremental materialization in the domain of stream reasoning were being proposed [4]. In particular, in [2], the authors proposed a novel algorithm and implemented it into the C-SPARQL execution engine. In

this algorithm all data structures are stored in the main memory. The evaluation of this approach, based on the transitive property, proved that it is faster than DRed for updates that involve less than 13% of changes into the knowledge base. However, this algorithm is tailored for a specific application scenario (stream reasoning in C-SPARQL), and relies on some strong assumptions, e.g., that the expiration time of triples is known a-priori. In practice, this only happens if triples are observed through a *fixed* time window; unpredictable changes or other kinds of observation windows are not supported.

8 Conclusions

In this paper, we presented *DynamiTE*, a parallel system designed to efficiently compute and maintain the materialization of a knowledge base in the presence of addition or removal of triples.

For data addition, *DynamiTE* implements a parallel version of the well-known semi-naive evaluation. For data removal, it implements two algorithms, one that is among the state of the art in the literature, and a more efficient one. *DynamiTE* is designed to exploit multi-core hardware for improved performance, adopting data structures that enable fast retrieval of information, e.g., to efficiently execute queries. Our evaluation shows the efficiency of *DynamiTE*, both in computing a complete materialization, and in managing incremental updates. Furthermore, it shows how the removal algorithm that we propose significantly outperforms existing state of the art approaches.

As future work, we plan to extend the algorithms implemented in *DynamiTE* to support different types of reasoning rules, and dynamic changes in the ruleset. Furthermore, future research could explore whether the implemented algorithms can be improved with heuristics. Finally, we intend to investigate how distributed processing can further increase the scalability.

To conclude, we have shown how our system encodes efficient parallel methods to perform full and incremental materialization adapting well-known algorithms to the task of reasoning. The throughput is higher than state of the art methods (per machine), so that a full materialization of 1 billion triples takes less than 75 minutes, while the response time to updates ranges from hundreds of milliseconds to a few minutes. This allows the system to perform a large-scale materialization on much more dynamic inputs than currently possible.

Acknowledgments. This research has been funded by the Dutch national program COMMIT.

References

1. Abiteboul, S., Hull, R., Vianu, V.: Foundations of Databases. Addison-Wesley (1995), `http://webdam.inria.fr/Alice/`
2. Barbieri, D.F., Braga, D., Ceri, S., Della Valle, E., Grossniklaus, M.: Incremental Reasoning on Streams and Rich Background Knowledge. In: Aroyo, L., Antoniou, G., Hyvönen, E., ten Teije, A., Stuckenschmidt, H., Cabral, L., Tudorache, T. (eds.) ESWC 2010, Part I. LNCS, vol. 6088, pp. 1–15. Springer, Heidelberg (2010)

3. Broekstra, J., Kampman, A.: Inferencing and Truth Maintenance in RDF Schema: Exploring a Naive Practical Approach. In: Workshop on Practical and Scalable Semantic Systems (PSSS), Sanibel Island, Florida (2003)
4. Della Valle, E., Ceri, S., van Harmelen, F., Fensel, D.: It's a streaming world! reasoning upon rapidly changing information. IEEE Intelligent Systems 24(6), 83–89 (2009)
5. Guo, Y., Pan, Z., Heflin, J.: LUBM: A benchmark for OWL knowledge base systems. Web Semantics: Science, Services and Agents on the World Wide Web 3, 158–182 (2005)
6. Gupta, A., Mumick, I.S.: Maintenance of Materialized Views: Problems, Techniques, and Applications. Data Engineering Bulletin 18(2), 3–18 (1995)
7. Gupta, A., Mumick, I.S., Subrahmanian, V.S.: Maintaining Views Incrementally. In: Proceedings of SIGMOD, vol. 22, pp. 157–166. ACM (1993)
8. Harrison, J.V., Dietrich, S.: Maintenance of Materialized Views in a Deductive Database: An update Propagation Approach. In: Workshop on Deductive Databases, JICSLP, pp. 56–65 (1992)
9. Hayes, P. (ed.): RDF Semantics. W3C Recommendation (2004)
10. Kolovski, V., Wu, Z., Eadon, G.: Optimizing Enterprise-scale OWL 2 RL Reasoning in a Relational Database System. In: Patel-Schneider, P.F., Pan, Y., Hitzler, P., Mika, P., Zhang, L., Pan, J.Z., Horrocks, I., Glimm, B. (eds.) ISWC 2010, Part I. LNCS, vol. 6496, pp. 436–452. Springer, Heidelberg (2010)
11. Kotowski, J., Bry, F., Brodt, S.: Reasoning as Axioms Change. In: Rudolph, S., Gutierrez, C. (eds.) RR 2011. LNCS, vol. 6902, pp. 139–154. Springer, Heidelberg (2011)
12. Munoz-Venegas, S., Prez, J., Gutierrez, C.: Simple and Efficient Minimal RDFS. Web Semantics: Science, Services and Agents on the World Wide Web 7(3) (2009)
13. Olson, M.A., Bostic, K., Seltzer, M.: Berkeley db. In: Proceedings of the FREENIX Track: 1999 USENIX Annual Technical Conference, pp. 183–192 (1999)
14. Prud'hommeaux, E., Seaborne, A.: SPARQL Query Language for RDF. W3C Recommendation (2008)
15. Staudt, M., Jarke, M.: Incremental Maintenance of Externally Materialized Views. In: Vijayaraman, T.M., Buchmann, A.P., Mohan, C., Sarda, N.L. (eds.) Proceedings of VLDB, pp. 75–86 (1996)
16. Urbani, J., Kotoulas, S., Maassen, J., Harmelen, F.V., Bal, H.: WebPIE: A Web-scale Parallel Inference Engine using MapReduce. Web Semantics: Science, Services and Agents on the World Wide Web 10, 59–75 (2012)
17. Urbani, J., Maassen, J., Drost, N., Seinstra, F., Bal, H.: Scalable RDF data compression with MapReduce. Concurrency and Computation: Practice and Experience 25(1), 24–39 (2013)
18. Volz, R., Staab, S., Motik, B.: Incremental Maintenance of Materialized Ontologies. In: Meersman, R., Schmidt, D.C. (eds.) CoopIS 2003, DOA 2003, and ODBASE 2003. LNCS, vol. 2888, pp. 707–724. Springer, Heidelberg (2003)
19. Volz, R., Staab, S., Motik, B.: Incrementally maintaining materializations of ontologies stored in logic databases. In: Spaccapietra, S., Bertino, E., Jajodia, S., King, R., McLeod, D., Orlowska, M.E., Strous, L. (eds.) Journal on Data Semantics II. LNCS, vol. 3360, pp. 1–34. Springer, Heidelberg (2005)
20. Weaver, J., Hendler, J.A.: Parallel Materialization of the Finite RDFS Closure for Hundreds of Millions of Triples. In: Bernstein, A., Karger, D.R., Heath, T., Feigenbaum, L., Maynard, D., Motta, E., Thirunarayan, K. (eds.) ISWC 2009. LNCS, vol. 5823, pp. 682–697. Springer, Heidelberg (2009)
21. Weiss, C., Karras, P., Bernstein, A.: Hexastore: Sextuple indexing for semantic web data management. In: Proceedings of VLDB, vol. 1, pp. 1008–1019 (2008)

Discovering Missing Semantic Relations between Entities in Wikipedia

Mengling Xu[1], Zhichun Wang[1,⋆], Rongfang Bie[1], Juanzi Li[2], Chen Zheng[1], Wantian Ke[1], and Mingquan Zhou[1]

[1] Beijing Normal University, Beijing, China
{zcwang,rfbie,mqzhou}@bnu.edu.cn,
{mengling,zc_cheney,kewantian}@mail.bnu.edu.cn
[2] Tsinghua University, Beijing, China
lijuanzi@tsinghua.edu.cn

Abstract. Wikipedia's infoboxes contain rich structured information of various entities, which have been explored by the DBpedia project to generate large scale Linked Data sets. Among all the infobox attributes, those attributes having hyperlinks in its values identify semantic relations between entities, which are important for creating RDF links between DBpedia's instances. However, quite a few hyperlinks have not been anotated by editors in infoboxes, which causes lots of relations between entities being missing in Wikipedia. In this paper, we propose an approach for automatically discovering the missing entity links in Wikipedia's infoboxes, so that the missing semantic relations between entities can be established. Our approach first identifies entity mentions in the given infoboxes, and then computes several features to estimate the possibilities that a given attribute value might link to a candidate entity. A learning model is used to obtain the weights of different features, and predict the destination entity for each attribute value. We evaluated our approach on the English Wikipedia data, the experimental results show that our approach can effectively find the missing relations between entities, and it significantly outperforms the baseline methods in terms of both precision and recall.

Keywords: Wikipedia, Infobox, Linked Data.

1 Introduction

Wikipedia is a free, collaborative, online encyclopedia that contains more than 20 million articles written in 285 languages by March 2013. Wikipedia articles contain rich structured information, such as infoboxes, categorization information, and links to external Web pages. Therefore, a number of projects have acquired data from Wikipedia to build large-scale machine readable knowledge bases [2,1,16,3]. One of the most valuable contents in Wikipedia is its infoboxes, which display articles' most important facts as a table of attribute-value pairs,

⋆ Corresponding author.

H. Alani et al. (Eds.): ISWC 2013, Part I, LNCS 8218, pp. 673–686, 2013.

and can be easily converted into machine-readable data. It was reported that DBpedia generated over 26 million RDF triples out of Wikipedia's infoboxes in 2009 by its generic infobox extraction algorithm. With the development of the DBpedia project these years, much more infobox RDF triples in 111 different languages have been generated.

Wikipedia uses infobox templates to define the schemas of infoboxes for different types of entities. An infobox template provides important attributes that are commonly used to describe related entities. Some attributes in infobox templates are relational that their values usually contain links referring to other entities within Wikipedia, which identify semantic relations between entities. Such relational attributes can be transformed into object properties in Linked Data, which facilitate establishing typed links between instances. Since creating links of structured data on the Web is the central idea of Linked Data, relational attributes in Wikipedia are especially important for creating Linked Data. However, sometimes the relational attributes cannot really connect entities in Wikipedia because the hyperlinks from the attributes' values to the corresponding entities are not annotated by editors. This problem causes lots of valuable relations between entities being missing in Wikipedia.

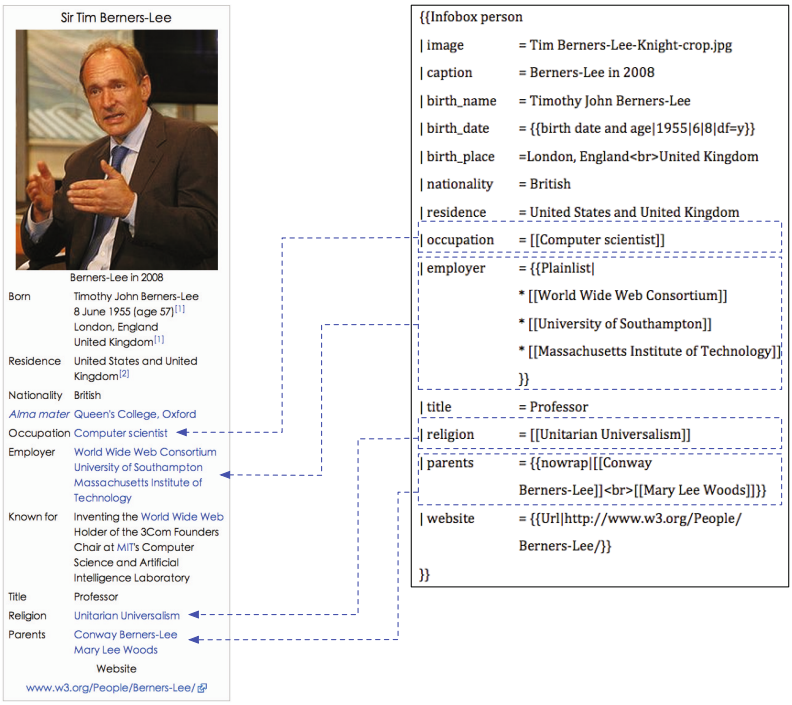
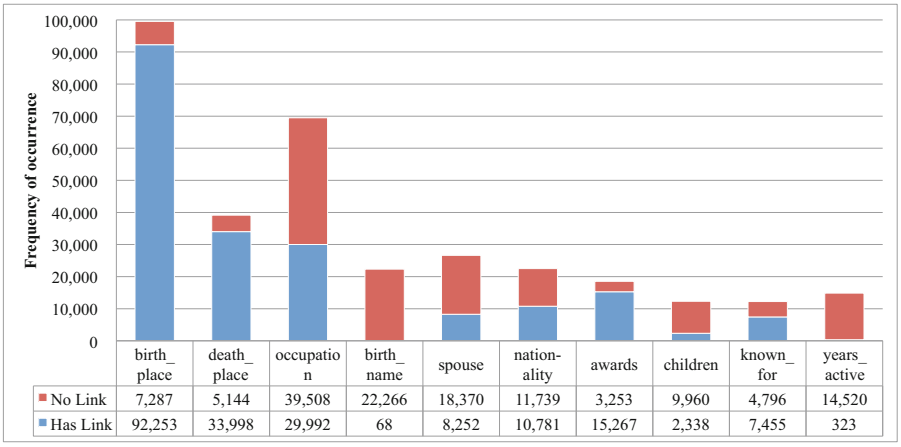


Fig. 1. Sample Wikipedia infobox: (a) display format; (b) editing format

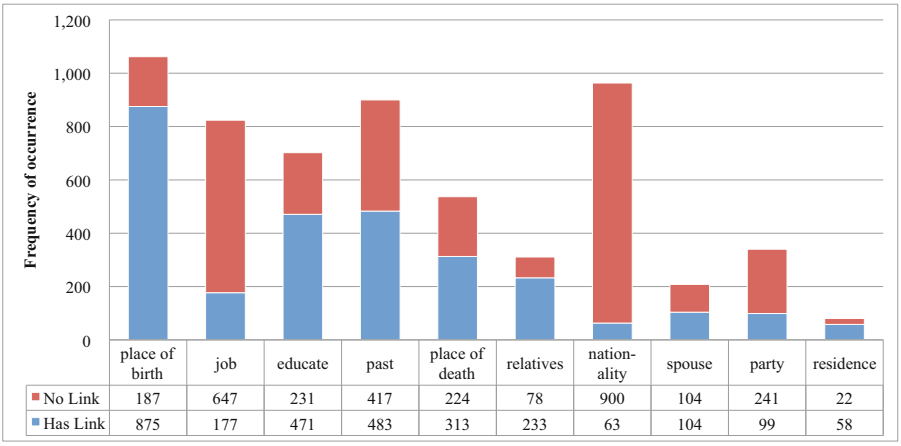
Fig. 1 (a) and (b) show a sample infobox and its source data in editing format from the article *Tim Berners-Lee* in Wikipedia, respectively. Relational attributes such as *Occupation* and *Parents* have values with links to other entities in Wikipedia; for these attributes, we use arrow lines to connect the corresponding contents in Fig. 1 (a) and (b). The attributes *Born_place*, *Nationality* and *Residence* are supposed to be relational, but there are no links in their values. This problem does not only occur in this sample infobox. In order to get insight into the entity links in the infoboxes, we investigate all the 123,246 English person infoboxes and 1,162 Chinese person infoboxes in Wikipedia. Fig. 2 and Fig. 3 show the number of times that the value has a link and has no link of the top ten frequently used attributes in English and Chinese, respectively. In Fig. 2, it is observed that most of the top used attributes (except for the attribute *birth_name* and attribute *years_active*) can be considered as relational ones because they contain large number of links in their values; however, there are still parts of their values having no links. The percentage of values without links varies among different attributes, which ranges form 7% (*birth_place* to 81% (*children*). Similar observations can also be obtained in Chinese infoboxes; Fig. 3 shows that larger portion of attribute values have no links comparing to English infoboxes.



	birth_ place	death_ place	occupation	birth_ name	spouse	nation- ality	awards	children	known_ for	years_ active
No Link	7,287	5,144	39,508	22,266	18,370	11,739	3,253	9,960	4,796	14,520
Has Link	92,253	33,998	29,992	68	8,252	10,781	15,267	2,338	7,455	323

Fig. 2. Statistics of links in person infoboxes in English Wikipedia

In order to solve the problem of missing semantic relations in Wikipedia, we need a system that can automatically add entity links in the attribute values in infoboxes. Recently, several approaches have been proposed to link entities in plain texts with Wikipedia [9,11,7,15]. These approaches first identify important named entities in the given text and then link them to the corresponding entities in Wikipedia. Since infoboxes contain structured information and are quite different from plain texts, traditional entity linking approaches cannot guarantee good results. Therefore, we propose an approach to automatically add entity links in infobox attribute values. Our approach first identifies all the entity mentions in a given infobox, and then decides the entity links based on 7 features of



	place of birth	job	educate	past	place of death	relatives	nation-ality	spouse	party	residence
No Link	187	647	231	417	224	78	900	104	241	22
Has Link	875	177	471	483	313	233	63	104	99	58

Fig. 3. Statistics of links in person infoboxes in Chinese Wikipedia

mention-entity pair. A learning model is used to obtain the appropriate weights of features, so that entity links can be predicted accurately.

The rest of this paper is organized as follows, Section 2 describes the proposed approach in detail; Section 3 presents the evaluation results; Section 4 discusses some related work and finally Section 5 concludes this work.

2 The Proposed Approach

In this section, we introduce our proposed approach in detail. Given an infobox with some missing entity links in their attribute values, our approach first automatically extracts the candidate name mentions that might refer to entities in Wikipedia, and then identifies the correct corresponding entity for each mention.

2.1 Mention Identification

To extract entity mentions in infoboxes, we build a mention dictionary that includes all the entity mentions in Wikipedia. In Wikipedia, an entity link is annotated by square brackets [[**entity**]] in the source data of articles. Here **entity** denotes the unique name of the referred entity. When the mentioned name of an entity is different from its unique name, the link is annotated by [[**entity** | **mention**]]; **mention** denotes the string tokens that actually appear in the text. In order to get all the mentions that have appeared in Wikipedia, we process all the annotated entity links in the form of [[**entity** | **mention**]] in Wikipedia. In addition, all the titles of articles in Wikipedia are also taken as mentions, which will be included in the mention dictionary. The mention dictionary also records the possible entities that each mention might refer to. Therefore, the dictionary can be represented as 2-tuple $D = (M, E)$, where $M = \{m_1, m_2, ..., m_k\}$ is the set of all mentions in Wikipedia, and $E = \{E_{m_1}, E_{m_2}, ..., E_{m_k}\}$ is the sets of entities corresponding to the mentions in M.

After the dictionary D being built, our approach extracts mentions in infoboxes by matching all the n-grams of the attribute values with mentions M in the dictionary D. The result of mention identification is a set of mentions that are matched by the n-grams. Because the goal of our approach is to find the missing entity links in infoboxes, only attribute values having no links will be processed to identify entity mentions.

2.2 Features for Predicting Entity Links

Once a set of mentions are identified in an infobox, our approach computes 7 features for each mention-entity pair to assess the possibility that a link exists between them from different respects.

Before defining other features, we first introduce a metric *Semantic Relatedness* [10]. This metric is used to compute the relatedness between the candidate entity and the context of a mention in different aspects.

Definition 1. ***Semantic Relatedness.*** *Given two entities a and b in Wikipedia, the Semantic Relatedness between a and b is computed as*

$$r(a,b) = 1 - \frac{log(max(|I_a|,|I_b|)) - log(|I_a \cap I_b|)}{log(|W|) - log(min(|I_a|,|I_b|))} \tag{1}$$

where I_a and I_b are the sets of inlinks of article a and article b, respectively; and W is the set of all articles in the input wiki.

Let B be a set of entities in Wikipedia, the Semantic Relatedness between an entity a and a set of entities B is defined as

$$SR(a,B) = \frac{1}{|B|}\sum_{b\in B} r(a,b) \tag{2}$$

Given a mention m in an infobox, let E_m represent the set of candidate entities that m might link to. For each entity $e \in E_m$, the following features are computed for (e,m).

Feature 1: ***Entity Occurrence***

According to the introduction of infobox given by Wikipedia, the information presented in the infobox should still be presented in the main text of the article. Therefore, if there is already a link to a certain entity in the text of article, there will be very likely a link to this entity in the infobox. Here, we define an *Entity Occurrence* feature to capture this information:

$$f_1(e,m) = \begin{cases} 1 \text{ if } e \in C_{article}(m) \\ 0 \text{ otherwise} \end{cases} \tag{3}$$

where $C_{article}(m)$ is the set of entities appearing in the main text of the current article containing m.

Feature 2: ***Link Probability***

Link Probability feature approximates the probability that a mention m links to an entity e:

$$f_2(e, m) = \frac{count(m, e)}{count(m)} \tag{4}$$

where $count(m, e)$ denotes the number of times that m links to e in the whole Wikipedia, and the $count(m)$ denotes the number of times that m appears in Wikipedia.

Feature 3: *Infobox Context Relatedness*
Let $C_{infobox}(m)$ be the set of entities already be linked in the infobox where m appear, we define the infobox context relatedness between a candidate entity $e \in E_m$ and a mention m as

$$f_3(e, m) = SR(e, C_{infobox}(m)) \tag{5}$$

Feature 4: *Article Context Relatedness*
Let $C_{article}(m)$ be the set of entities already linked by mentions in the article text that m appear, we define the article context relatedness between a candidate entity $e \in E_m$ and mention m as

$$f_4(e, m) = SR(e, C_{article}(m)) \tag{6}$$

Feature 5: *Abstract Context Relatedness*
The first paragraph in the text of an article usually defines the subject of the article, and contains the most important information about the subject of the article, which is usually called the abstract or the definition of the article. Let $C_{abstract}(m)$ be the entities appear in the abstract, here we define the abstract context relatedness between a candidate entity $e \in E_m$ and a mention m as

$$f_5(e, m) = SR(e, C_{abstract}(m)) \tag{7}$$

Feature 6: *Attribute Range Context Relatedness*
Let $C_{att\_rang}(m)$ be the set of entities that appear in the value of attribute att_m, we define the attribute value context relatedness between a candidate entity $e \in E_m$ and mention m as

$$f_6(e, m) = SR(e, C_{att\_rang}(m)) \tag{8}$$

Attribute Range Context Relatedness can assess the similarity between a candidate entity and the set of entities that have already been linked in the value of a concerned attribute. Therefore, this feature can estimate what types of entities are more likely to be linked by the concerned attribute.

Feature 7: *Attribute Domain Context Relatedness*
Let $C_{att\_dom}(m)$ be the set of entities that described by the attribute att_m, we define the attribute domain context relatedness between a candidate entity $e \in E_m$ and mention m as

$$f_7(e, m) = SR(e, C_{att\_dom}(m)) \tag{9}$$

2.3 Learning to Predict New Entity Links

To predict new entity links, our approach computes the weighted sum of features between mentions and entities by the following score function:

$$s(m,e) = \omega_1 \times f_1(m,e) + ... + \omega_6 \times f_6(m,e) + \omega_7 \times f_7(m,e) \tag{10}$$

For each mention m, the entity e^* that maximizes the score function $s(m,e^*)$ is predicted as the destination entity of m. The idea of predicting entity links is simple and straight, but how to appropriately set the weights of different similarity features is a challenging problem, which highly influences the final results.

Here, we use the already existing entity links $L = \{< m_i, e_i >\}_{i=1}^{k}$ in infoboxes as training data, and train a logistic regression model to get the weights of different features. Given a mention m and its corresponding entity e, the learned weights should ensure

$$\boldsymbol{\omega} \cdot (\boldsymbol{f}(m,e^*) - \boldsymbol{f}(m,e)) > 0, (e \in E_m, e \neq e^*) \tag{11}$$

where $\boldsymbol{\omega} =< \omega_1, ..., \omega_7 >$ and $\boldsymbol{f}(\cdot) =< f_1(\cdot), ..., f_7(\cdot) >$.

Therefore, we can use the sigmoid function to compute the probability that an entity e_1 is better than another entity e_2 (denoted as $e_1 \succ e_2$) as the destination for a mention.

$$P((e_1 \succ e_2) = true) = \frac{1}{1 + e^{-\boldsymbol{\omega} \cdot (\boldsymbol{f}(m,e_1) - \boldsymbol{f}(m,e_2))}} \tag{12}$$

If $s(m,e_1) > s(m,e_2)$, $P((e_1 \succ e_2) = true) > 0.5$; otherwise $P((e_1 \succ e_2) = true) < 0.5$. In this case, the weights $\boldsymbol{\omega}$ can be determined by the MLE (maximum likelihood estimation) technique for logistic regression.

Therefore, we generate a new dataset $D = \{(\boldsymbol{x_j}, y_j)\}_{j=1}^{m}$ based on the known entity links $L = \{< m_i, e_i >\}_{i=1}^{k}$ to train a logistic regression model; $\boldsymbol{x_j}$ is the input vector and y_j represents the class label (positive or negative). For each mention m_i, a positive example $(\boldsymbol{f}(m_i,e_i) - \boldsymbol{f}(m_i,e^{'}), positive)$ or a negative example $(\boldsymbol{f}(m_i,e^{'}) - \boldsymbol{f}(m_i,e_i), negative)$ is generated for each entity $e^{'} \in (E_{m_i} - \{e_i\})$. We make the number of positive examples and negative examples be the same, which avoids the imbalanced classification problem. After the logistic regression model being trained, the learned weights $\boldsymbol{\omega} =< \omega_1, ..., \omega_7 >$ will be used in Equation 10 to predict new entity links.

For some identified mentions, there might not be its corresponding entities in Wikipedia. Therefore, a threshold δ is set to filter out entity links with low scores. In the learning process, when the optimal weights of features are obtained, the threshold δ is determined by optimizing the overall performance on the training dataset.

3 Experiment

3.1 Datasets

We use the datasets of English Wikipedia to evaluate the proposed approach. We downloaded the English Wikipedia XML dump from Wikipedia's download site[1], which was archived in August 2012, and has 4 million articles. 100 infoboxes are randomly chosen from the whole dataset for the evaluation. There are 630 already existing entity links in the selected infoboxes, 50% of these links are randomly selected as the ground truth for the evaluation, which are removed from the infoboxes before the infoboxes are fed to our approach. After the execution of our approach, we collect the new discovered entity links and compare them against the ground truth links. In the experiments, 40% of the selected ground truth links were used for training the prediction model, the rest of 60% selected links were used as testing data in the evaluation.

3.2 Evaluation Metrics

We use precision, recall, and F1-score to evaluate the performance of the proposed approach. These measures are computed as follows:

Precision (p): It is the percentage of correctly discovered entity links in all the discovered entity links.

$$p = \frac{|A \cap T|}{|A|} \tag{13}$$

where T is the set of ground truth entity links, A is the set of discovered entity links.

Recall (r): It is the percentage of correctly discovered entity links in the ground truth entity links.

$$r = \frac{|A \cap T|}{|T|} \tag{14}$$

F1-score ($F1$): F1-Measure considers the overall result of precision and recall.

$$F1 = \frac{2pr}{p + r} \tag{15}$$

3.3 Comparison Methods

Here we use three comparison methods as the baselines of evaluation:

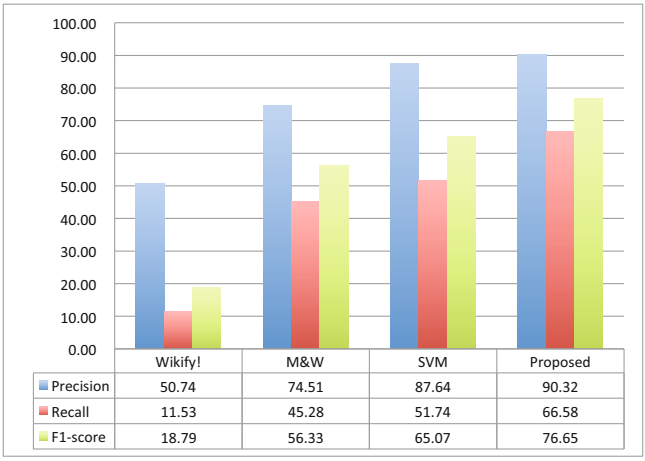
- **Wikify!.** This method was proposed by Mihalcea and Csomai [9], which is able to automatically perform the annotation task following the Wikipedia guidelines. Wikify! first uses a unsupervised extraction algorithm to identify and rank mentions, and then combines both knowledge-based approach and data-driven method to discover new entity links.

[1] `http://dumps.wikimedia.org/enwiki/`

- **M&W.** Milne and Witten proposed an learning based entity linking approach [11]. Their approach uses three features (Commonness, Relatedness, and Context Quality) and C4.5 classifier to predict new entity links. Here, we first use our approach to identify mentions in the infoboxes, and then employ Milne and Witten's disambiguation method to predict new entity links.
- **SVM.** This method first computes the similarities defined in Section 2.2 for each mention-entity pair, and then trains a SVM [4] classification model on the training entity links. New mention-entity pairs are predicted by the trained SVM as entity links or not entity links.

3.4 Results Analysis

Here we first compare the performance of our approach with the comparison methods, and then analysis the contribution of different features in our approach.



	Wikify!	M&W	SVM	Proposed
Precision	50.74	74.51	87.64	90.32
Recall	11.53	45.28	51.74	66.58
F1-score	18.79	56.33	65.07	76.65

Fig. 4. Performance of different methods (%)

Performance Comparison. Fig. 4 shows the performance of 4 different methods. According to the results, the Wikify! method does not perform very well on the infobox data. Wikify! only achieves 50.74% precision and 11.53% recall. It seems that Wikify! can not make good decision given only the string tokens of a infobox. The method of M&W performs better than Wikify!, but the SVM method achieves both better precision and recall than M&W. Therefore, it shows that the features defined for our approach have better discriminant ability than the features in M&W method. Compared with three baseline methods, our proposed approach achieves the best results in terms of both precision and recall. Our proposed approach outperforms SVM method by 11.58% in terms of F1-score, which means that the learning method in our approach is more suitable for the entity linking tasks; training classifiers directly on the original features cannot get the best performance.

Feature Contribution Analysis. Among 7 defined features, which one is the most important? To get insight to this question, we perform an analysis on the contribution of different features. Here, we run our approach 7 times on the evaluation data. Each time one feature is removed from the feature vectors of mention-entity pairs. We record the decrease of F1-score for each feature when it is removed; it is reasonable to evaluate the importance of each feature by comparing their corresponding F1-score decrease. Fig. 5 compares the importance of different features. According to the results, we can rank these features based on their importance in a descending order as: Feature 1, Feature 6, Feature 3, Feature 5, Feature 2 and Feature 7.

It seems that the occurrence of candidate entities in the main text of article is very important for identifying the correct entity links. The *Attribute Range Context Relatedness* feature is also important, it might because this feature can reflect what types of entities are possible to appear in the values of certain attributes. The *Attribute Domain Context Relatedness* feature is the least important one among all the 7 features, it might because different entities usually have different values of the same attribute; the relatedness between candidate entities and the entities described by a specific attribute is less relevant to the entity links.

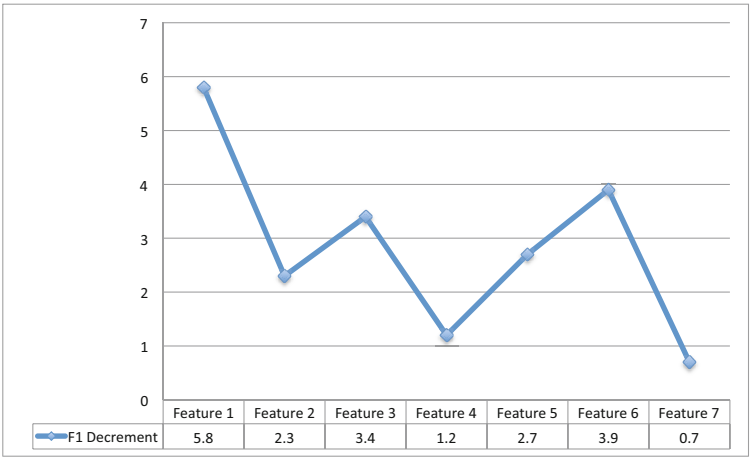


Fig. 5. Contribution analysis of different features (%)

4 Related Work

In this section, we review some related work.

4.1 Entity Linking

A group of closely related work is *Entity Linking*, which aims to identify entities in documents and link them to a knowledge base, such as Wikipedia and DBpedia.

Wikify! [9] is a system which is able to automatically perform the annotation task following the Wikipedia guidelines. Wikify! has two components: the keyword extraction and the link disambiguation. In the first components, Wikify! uses a unsupervised keyword extraction algorithm to identify and rank mentions. In the disambiguation component, Wikify! combines both knowledge-based approach and data-driven method to predict the links from mentions to entities in Wikipedia.

Milne et al. [11] proposed a learning based approach for linking entities in text to Wikipedia. Their approach trains a C4.5 classifier based on three features (commonness, relatedness and context quality) of entity-mention pairs for link disambiguation. A classification algorithm is also used in the candidate link detection.

Kaulkarni et al. [7] proposed a collective approach for annotating Wikipedia entities in Web text. Their approach differs from the former approaches in that it combines both local mention to entity compatibility and global document level topical coherence. The collective prediction of entity links improves the accuracy of results.

Following a similar collective decision idea, Han et al. [6] proposed a graph-based collective entity linking algorithm. Their approach first construct a referent graph, where nodes corresponds to all name mentions in a document and all possible referent entities of these name mentions, edge between a name mention and an entity represents a compatible relation between them, edge between two entities represents a sematic-related relation between them. Both the compatibility and semantic relatedness are propagate through the referent graph. The entity linking problem is solved by selecting the entity for a mention that maximizes the product of compatibility and relatedness.

Mendes et al. [8] developed a system DBpedia Spotlight for automatically annotating text documents with DBpedia URIs. DBpedia Spotlight first recognizes the phrases in a sentence that may indicate a mention of a DBpedia entity; then the recognized mention is mapped to candidate entities in DBpedia; a disambiguation stage is employed to find the most likely entities for the mention. The disambiguation task is cast as a ranking problem in DBpdia Spotlight, and Vector Space Model and a new weighting method Inverse Candidate Frequencty (ICF) are used for similarity computation.

Shen et al. [15] proposed a system LINDEN, which is a novel framework to link named entities in text with a knowledge base by leveraging the rich semantic knowledge embedded in the Wikipedia and the taxonomy of the knowledge base. The LINDEN builds a feature vector for each entity, which includes link probability, semantic associativity, semantic similarity and global coherence. And the system uses a max-margin technique to rank the candidate entities for the entity mentions.

LIEGE [14] is another work of Shen et al., a general framework for linking entities in web lists with knowledge base. In order to find the proper entity from knowledge base as the mapping entity for the list item, LIEGE defines several metrics to measure the link quality of the candidate mapping entity,

including the prior probability, coherence, type hierarchy based similarity, and distributional context similarity. A max-margin technique is used to learn the weights for different feature values to calculate the linking quality.

The above entity linking approaches mainly take plain texts as inputs, and the infoboxes are quite different from plain texts. Information in infoboxes is structured, and some existing entity links might appear in infoboxes. Based on these observations, we define more specific features to describe the relations between mentions and entities. What's more, we use a new learning method to get the weights of different features. Because there are lots of context entity links for each mention in infobox, we can still get desired results when entity links are predicted not in a collective way.

4.2 Instance Matching

Another group of related work is *Instance Matching*, which aims to find equivalent entities in different linked datasets. Instance matching tools can be used to find new RDF links between linked datasets.

Silk [17] is a link discovery engine which automatically finds RDF links between data sets. Users must specify which type of RDF links should be discovered between the data sources as well as which conditions data items must fulfill in order to be interlinked. These link conditions can apply different similarity metrics to multiple properties of an entity or related entities that are addressed using a path-based selector language.

idMesh [5] is a graph-based algorithm for online entity disambiguation based on a probabilistic graph analysis of declarative links relating pairs of entities. idMesh derives a factor-graph from the entity and the source graphs to retrieve equivalent entities.

Raimond et al.[13] propose a interlinking algorithm for automatically linking music-related data sets on the web, taking into account both the similarities of the web resources and of their neighbors. Their algorithm provides online linking function based on accessing data through SPARQL end-points.

Nikolov et al. [12] present a data integration architecture called KnoFuss and proposed a component-based approach, which allows flexible selection and tuning of methods and takes the ontological schemata into account to improve the reusability of methods.

The purpose of our approach is to add semantic relations between entities in Wikipedia, which will finally enrich RDF links in DBpedia. Although *Instance Matching* also can add RDF links between different datasets, it mainly focuses on discovering the *sameAs* links, which is different from finding arbitrary typed relations between entities.

5 Conclusion and Future Work

In this paper, we propose an approach for automatically discovering the missing typed relations between entities in Wikipedia's infoboxes. Our approach works

in two steps: it first identifies entity mentions in the given infoboxes, and uses a learning model to predict new entity links based on several features of mention-entity pair. The experimental results show that our approach can accurately find missing links in infoboxes, and it performs better than the baseline methods.

Actually, there are some wrongly annotated entity links in Wikipedia's infoboxes. Besides of adding new entity links in infoboxes, we also want to discover wrong entity links in infoboxes in the future work. By the efforts of adding and refining entity links in infoboxes, more semantic relations of high quality between entities can be obtained. Our future work also includes extending our approach to solve the problem of finding missing RDF links between linked datasets.

Acknowledgement. The work is supported by NSFC (No. 61202246, 61035004, 61171014), NSFC-ANR(No. 61261130588), 863 High Technology Program (2011AA01A207), FP7-288342, THU-NUS NExT Co-Lab, and the Fundamental Research Funds for the Central Universities.

References

1. Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., Ives, Z.G.: DBpedia: A nucleus for a web of open data. In: Aberer, K., et al. (eds.) ASWC 2007 and ISWC 2007. LNCS, vol. 4825, pp. 722–735. Springer, Heidelberg (2007)
2. Bizer, C., Lehmann, J., Kobilarov, G., Auer, S., Becker, C., Cyganiak, R., Hellmann, S.: DBpedia - a crystallization point for the web of data. Web Semantics: Science, Services and Agents on the World Wide Web 7(3), 154–165 (2009)
3. Bollacker, K.D., Cook, R.P., Tufts, P.: Freebase: a shared database of structured general human knowledge. In: Proceedings of the 22nd National Conference on Artificial Intelligence, vol. 2, pp. 1962–1963 (2007)
4. Cortes, C., Vapnik, V.: Support-vector networks. Machine Learning 20(3), 273–297 (1995)
5. Cudre-Mauroux, P., Haghani, P., Jost, M., Aberer, K., De Meer, H.: idMesh: graph-based disambiguation of linked data. In: Proceedings of the 18th International Conference on World Wide Web, pp. 591–600 (2009)
6. Han, X., Sun, L., Zhao, J.: Collective entity linking in web text: a graph-based method. In: Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 765–774 (2011)
7. Kulkarni, S., Singh, A., Ramakrishnan, G., Chakrabarti, S.: Collective annotation of wikipedia entities in web text. In: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 457–466 (2009)
8. Mendes, P.N., Jakob, M., García-Silva, A., Bizer, C.: DBpedia spotlight: shedding light on the web of documents. In: Proceedings of the 7th International Conference on Semantic Systems, pp. 1–8 (2011)
9. Mihalcea, R., Csomai, A.: Wikify!: linking documents to encyclopedic knowledge. In: Proceedings of the Sixteenth ACM Conference on Conference on Information and Knowledge Management, pp. 233–242 (2007)
10. Milne, D., Witten, I.H.: An effective, low-cost measure of semantic relatedness obtained from wikipedia links. In: Proceedings of the First AAAI Workshop on Wikipedia and Artificial Intelligence (2008)

11. Milne, D., Witten, I.H.: Learning to link with wikipedia. In: Proceedings of the 17th ACM Conference on Information and Knowledge Management, pp. 509–518 (2008)
12. Nikolov, A., Uren, V.S., Motta, E., Roeck, A.N.D.: Handling instance coreferencing in the knofuss architecture. In: 1st International Workshop on Identity and Reference on the Semantic Web (2008)
13. Raimond, Y., Sutton, C., Sandler, M.: Automatic interlinking of music datasets on the semantic web. In: Proceedings of the 1st Linked Data on the Web Workshop
14. Shen, W., Wang, J., Luo, P., Wang, M.: LIEGE: link entities in web lists with knowledge base. In: Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1424–1432 (2012)
15. Shen, W., Wang, J., Luo, P., Wang, M.: LINDEN: linking named entities with knowledge base via semantic knowledge. In: Proceedings of the 21st International Conference on World Wide Web, pp. 449–458 (2012)
16. Suchanek, F.M., Kasneci, G., Weikum, G.: YAGO: a core of semantic knowledge. In: Proceedings of the 16th International Conference on World Wide Web, pp. 697–706 (2007)
17. Volz, J., Bizer, C., Gaedke, M., Kobilarov, G.: Discovering and maintaining links on the web of data. In: Bernstein, A., Karger, D.R., Heath, T., Feigenbaum, L., Maynard, D., Motta, E., Thirunarayan, K. (eds.) ISWC 2009. LNCS, vol. 5823, pp. 650–665. Springer, Heidelberg (2009)

Infrastructure for Efficient Exploration of Large Scale Linked Data via Contextual Tag Clouds

Xingjian Zhang, Dezhao Song, Sambhawa Priya, and Jeff Heflin

Dept. of Computer Science and Engineering, Lehigh University
19 Memorial Drive West, Bethlehem, PA 18015, USA
{xiz307,des308,sps210,jeh3}@lehigh.edu

Abstract. In this paper we present the infrastructure of the contextual tag cloud system which can execute large volumes of queries about the number of instances that use particular ontological terms. The contextual tag cloud system is a novel application that helps users explore a large scale RDF dataset: the tags are ontological terms (classes and properties), the context is a set of tags that defines a subset of instances, and the font sizes reflect the number of instances that use each tag. It visualizes the patterns of instances specified by the context a user constructs. Given a request with a specific context, the system needs to quickly find what other tags the instances in the context use, and how many instances in the context use each tag. The key question we answer in this paper is how to scale to Linked Data; in particular we use a dataset with 1.4 billion triples and over 380,000 tags. This is complicated by the fact that the calculation should, when directed by the user, consider the entailment of taxonomic and/or domain/range axioms in the ontology. We combine a scalable preprocessing approach with a specially-constructed inverted index and use three approaches to prune unnecessary counts for faster intersection computations. We compare our system with a state-of-the-art triple store, examine how pruning rules interact with inference and analyze our design choices.

Keywords: Linked Data, Tag Cloud, Semantic Data Exploration, Scalability.

1 Introduction

We present the contextual tag cloud system[1] as an attempt to address the following questions: How can we help casual users explore the Linked Open Data (LOD) cloud? Can we provide a more detailed summary of linkages beyond the LOD cloud diagram[2]? Can we help data providers find potential errors or missing links in a multi-source dataset of mixed quality? There are two aspects of a dataset: the ontological terms (classes and properties) and the instances; and correspondingly, there are two types of linkages: ontological alignment and

[1] Contextual Tag Cloud Browser. http://gimli.cse.lehigh.edu:8080/btc/
[2] The Linking Open Data cloud diagram. http://lod-cloud.net/

H. Alani et al. (Eds.): ISWC 2013, Part I, LNCS 8218, pp. 687–702, 2013.

`owl:sameAs` links between instances. We allow the user to specify a *context* as a combination of ontological terms, and then visualize the degree of overlap between this context and all other terms. The context can be thought of as a class expression in description logic, but is significantly simplified for usability reasons. The overlap is the intersection of the context class and any other term. An appropriate visualization of these counts can reflect the patterns of co-occurrence of ontological terms as used in the instance data.

We build on the idea of a contextualized tag cloud system. In analogy to traditional Web 2.0 tag cloud systems, an instance is like a web document or photo, but is "tagged" with formal ontological classes, as opposed to folksonomies. Thus, we simply use "tags" as another name for the categories of instances. We extend the expressiveness and treat classes, properties and inverse properties as tags that are assigned to any instances using these ontological terms in their triples. The font sizes in the tag cloud reflect the number of matching instances for each tag. To explore the data, users can select a set of tags to form a context and the displayed tags are resized to indicate intersection with this context. Note, this system is neither an information retrieval system nor a SPARQL query engine, instead it is designed for exploration and pattern discovery.

With any uncurated dataset, one must maintain a healthy skepticism towards all axioms. Although materialization can lead to many interesting facts, a single erroneous axiom could generate thousands of errors. Rather than attempting to guess which axioms are worthwhile, our system supports multiple levels of inference; and at any time a user can view tag clouds with the same context under different entailment regimes, which helps users understand the dataset better and helps data providers investigate the errors in the dataset.

These simple but powerful interface concepts propelled the Contextual Tag Cloud Browser to win the Billion Triples Track of the 2012 Semantic Web Challenge[3]. Our initial version of the system [17] was used on DBPedia data [3]. For the Semantic Web Challenge, we added features and loaded the entire 2012 BTC dataset. This complex dataset contains 1.4 billion triples, from which we extract 198.6M unique instances, and assign more than 380K tags to these instances. This multi-source, large-scale dataset brings us challenges in achieving acceptable performance and user-interface design. Although we believe the user interface provides a convenient tool for exploring a Linked Data dataset, the focus of this paper is presenting novel approaches for efficient and scalable computation over noisy data with tremendous diversity.

The contributions of this paper are: (1) We propose using an inverted index to speed up a special kind of query, namely querying the intersection of generalized classes, and propose a scalable approach to preprocess it; (2) Some special cases of these queries can be answered without accessing the index, we propose three approaches to prune unnecessary queries and analyze alternative preprocessing approaches; (3) We develop formulae for supporting the first problems with multi-level inference and discuss our decision to materialize entailments and an efficient mechanism to store the results of these entailments. Although this

[3] SWC 2012 Winners. `http://challenge.semanticweb.org/2012/winners.html`

paper focuses on a very specific application, we believe scalable computation of conditional distributions can be applied to statistic based algorithms such as association rule learning. The rest of the paper is organized as follows: we first briefly describe the use cases of the tag cloud system and formally define the problem; then we discuss the preprocessing and online computation and how we support multi-level inference; after that we provide some experimental results of the system; then we compare with related works; and lastly we conclude.

2 The Problem: Use Case and Formal Definition

Initially, the system shows a tag cloud with no context tags selected, and the tags in the cloud reflect the number of instances related to each tag. If a tag is clicked, it will be added to the current context, and then a new tag cloud will be shown for the updated context. A user can add/remove any tags to/from the context, and explore any dynamically defined types of instances specified by the context. Then in the resulting tag cloud, the font size for each tag reflects the number of instances possessing the tag within the type specified by the contexts. Mathematically, this contextual tag cloud actually reveals the **conditional distribution** of the data: the probability that an instance has a tag given that it is an instance of the user-defined type. For example in Fig. 1, the property tag cloud shows us the degree to which instances of `foaf:Group` that are not in `schema:MusicGroup` are used with specific properties.

This kind of pattern visualization helps users learn about the dataset for different purposes. For example, large tags indicate frequent co-occurrence and can be used to form a SPARQL query spanning diverse, multiple, linked data

Fig. 1. Property Tag Cloud with context `foaf:Group` and ~`schema:MusicGroup`

sources that is most likely to return results; by focusing on the smaller tags, users can investigate rare combinations, and by drilling into the data determine if these are unusual facts or the product of data errors, such as incorrect `owl:sameAs` links. Additionally the user can dynamically change which entailment regime will be used to generate the tag cloud, thereby getting a big picture view of the impact of entailment on the data. This feature can be used to track down schema errors such as incorrect `rdfs:domain` statements.

An important aspect of the user interface is responsiveness. Ideally, each new tag cloud should be generated in under one second, or users will quickly doubt the system and/or become bored. Achieving this goal is particularly challenging since the dataset contains billions of triples with hundreds of thousands of ontological tags. In addition to an interface design that ensures the user is presented with partial results as quickly as possible, we carefully designed an infrastructure that is optimized for our unique form of queries. Before we describe our approach, we now formalize the computation problem.

Given a RDF dataset, an entailment regime R defines what kind of entailment rules will be applied to the explicit triples. In our implementation, we have two specific sets of rules: R_{Sub} for sub/equivalent class/property entailment (`rdfs5`, `rdfs7`, `rdfs9` and `rdfs11`[4]); and R_{DR} for property domain/range entailment (`rdfs2`, `rdfs3`). We also support the combination of these two sets, leading to four distinct entailment regimes $\mathcal{R} = \{\emptyset, R_{Sub}, R_{DR}, R_{Sub} \cup R_{DR}\}$.

Let $\mathcal{I}$ be the set of all the instances, and $\mathcal{T}$ be the set of all possible tags assigned to instances in the dataset. Given R, we define a function $\text{Tags}_R : \mathcal{I} \rightarrow 2^{\mathcal{T}}$ that returns all the tags assigned to the given instance under R-inference closure. For $i \in \mathcal{I}$ we assign three types of tags: (1) **Class** C, if $\langle i,$ `rdf:type`, $C\rangle$ is entailed under R. (2) **Property** p, if $\exists j \in \mathcal{I}$, s.t. $\langle i, p, j\rangle$ is entailed. (3) **Inverse Property** $p-$, if $\exists j \in \mathcal{I}$, $\langle j, p, i\rangle$ is entailed. Note under monotonic logic, $R_1 \subseteq R_2 \Rightarrow \text{Tags}_{R_1}(i) \subseteq \text{Tags}_{R_2}(i)$. The function $\text{Inst}_R : 2^{\mathcal{T}} \rightarrow 2^{\mathcal{I}}$ returns the set of all instances assigned the given set of tags. For convenience, we define the frequency of a set of tags T as $f_R(T) = |\text{Inst}_R(T)|$.

We can generalize various entailment rules into tag subsumptions. Tag t_1 is a sub tag of tag t_2 if and only if the entailment regime requires $\text{Inst}_R(\{t_1\}) \subseteq \text{Inst}_R(\{t_2\})$. This sub tag relation includes RDF subclasses/subproperties plus the ones entailed by the domain/range axioms: If $\langle p,$ `rdfs:domain`, $C\rangle$ and $\langle p,$ `rdfs:range`, $D\rangle$, then p is a sub tag of C and $p-$ is a sub tag of D. We compute the full closure on tag subsumptions for each inference regime.

A context is an expression of tags dynamically constructed by a user. In our implementation, it is the intersections of any number of tags or their negations. A **Negation Tag** $\sim t$ is virtually assigned to an instance i, if $t \notin \text{Tags}_R(i)$. Note that the semantics are based on negation-as-failure. We argue that this is the correct semantics for a system where what is not said is sometimes as important as what is said. Thus a context with $\{t_1, \ldots, t_n, \sim s_1, \ldots, \sim s_m\}$ actually defines a subset of instances: $\text{Inst}_R(\{t_1, \ldots, t_n\}) - \bigcup_{x=1,\ldots,m} \text{Inst}_R(\{s_x\})$. For a given context and entailment regime R, the system shows all the tags used by any

[4] RDFS Entailment Rules: `http://www.w3.org/TR/rdf-mt/#RDFSRules`

instance in the subset specified by the context, and the size of each tag reflects the number of instances having this tag within the subset.

Due to limited space, we omit the subtle details required to process negation tags for the remainder of this paper. This allows us to present a simplified exposition where a context $T \subset \mathcal{T}$ is a set of tags, and the instances specified by the context is $\text{Inst}_R(T)$. For more details, please refer to our technical report [18].

Since our goal is to display the frequency of all tags given a context T, our main challenge is to compute $f_R(\{t\} \cup T)$ for $\forall t \in \mathcal{T}$ efficiently. There are two ways to approach this problem: (1) ensure efficient calculation of $f_R(T)$ for any T; and (2) prune unnecessary calls of $f_R(\{t\} \cup T)$. To achieve this, we need to correctly structure the repository and develop an efficient preprocessing step. In the following section we will solve these problems for the situation where there is only a single set of inference rules R. Then we will discuss how to "infer" relations between tags and instances, and how to determine co-occurrence between tags under tag inference.

3 Preprocessing

Our previous experiments [17] showed that an RDBMS with decomposed storage model [1,11] is not as efficient as using an Information Retrieval (IR) style index for this specific application purpose, both in terms of load time (8X slower) and online query time (18X slower). Therefore we extend our IR approach, but meanwhile add more steps to deal with the BTC dataset.

Our preprocessing is shown in Figure 2, where the dashed boxes are input or intermediate data and the solid ones are data results for the online system. First, we parse the raw data and categorize triples into three files: the ontology file which includes specific properties (e.g., `rdfs:subClassOf`) or classes (e.g., `owl:Class`), the `owl:sameAs` (instance equivalence statements) file, and the file of remaining instance triples. This step can be skipped if the ontology and sameAs files are provided separately. The ontology is processed into a closure set of sub-tag axioms for the given entailment regime (or regimes); the closure is then responsible for two functions: $\text{sub}_R(t)$ and $\text{super}_R(t)$ which respectively return the sets of sub/super tags of tag t under inference R. We also use the well-known union-find algorithm to compute the closure for `owl:sameAs` statements, and pick a canonical id for each `owl:sameAs` cluster.

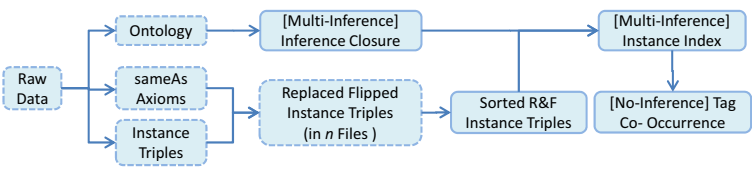


Fig. 2. Preprocessing for the tag cloud system

Then for the instance triples, we replace each instance with its `owl:sameAs` canonical id (if any). If the object of the triple is also an instance, we flip the triple and add it to the intermediate file, i.e., if the triple is $\langle i, p, j \rangle$, the flipped one is $\langle j, p-, i \rangle$. By this means, we can find all the tags (particularly inverse property tags) of an instance i by simply looking at the triples with i as a subject. In order to index an instance, we need to first group all of its triples together. To do this, we first output the triples into n files based on the hashcode of their subjects, so that we keep the information of an instance in the same file while making each file relatively small. Then we use merge sort on each "replaced and flipped" file, so that triples with the same subject instance are clustered together. Note that by splitting the triples into n files, we gain benefits from two sides: (1) sorting each file becomes faster (and since we only need to group triples with the same subject, we do not need to merge the sorted files); (2) we can sort in parallel (either with multiple machines or with multiple threads). We use these sorted files together with the given inference closure to build an inverted index of the instances.

The inverted index is built with tags as indexing terms and each tag has a sorted posting list of instances with that tag. This means given a "type" defined by a set of tags we can quickly find all the instances by doing an intersection over the posting lists. Also, since we use negation as failure, we do not need to index negation tags; their size can be calculated from its complementary tag. i.e. $f_R(\{\sim t\} \cup T) = f_R(T) - f_R(\{t\} \cup T)$. Meanwhile we add other fields such as labels of instances, sameAs sets, file pointers to the raw file, etc. to facilitate other features in our tag cloud system.

To help prune unnecessary tags when computing the conditional distribution of tags under any given context T, we precompute the **Co-occurrence Matrix** for all the tags. Define M_R as a $|\mathcal{T}| \times |\mathcal{T}|$ symmetric boolean matrix, where $M_R(x, y)$ denotes whether tags t_x and t_y co-occur, i.e. $M_R(x, y) = (f_R(\{t_x, t_y\}) > 0)$. There are three ways to generate M_R.

1. **Traverse all the instances.** For each instance $i \in \mathcal{I}$, we get all of its tags $\text{Tags}_R(i)$, for any pair of tags $(t_x, t_y) \in \text{Tags}_R(i) \times \text{Tags}_R(i)$, set $M_R(x, y)$.
2. **Traverse pairs of tags.** For any pair of tags $(t_a, t_b) \in \mathcal{T} \times \mathcal{T}$, if $f_R(\{t_a, t_b\}) > 0$, set $M_R(x, y)$.
3. **Traverse tag instances.** For each tag $t_x \in \mathcal{T}$, we get all of its instances $\text{Inst}_R(\{t_x\})$, and then set occurrences for all tags in them. For $i \in \text{Inst}_R(t_x)$, for any tag $t_y \in \text{Tags}_R(i)$, set $M_R(x, y)$.

We can roughly estimate the execution time of each method from how much index access (the functions Tags_R, f_R, and Inst_R) is needed. Assume on average a tag has d instances and an instance has e tags. The cost of $\text{Inst}_R(\{t_x, t_y\})$ (or $f_R(\{t_x, t_y\})$) is estimated as $c_1 d$, because the intersection needs to simultaneously walk through both sorted posting lists. The cost of $\text{Tags}_R(i)$ is estimated as $c_2 e$. Here, c_1, c_2 are constants given the dataset and the environment. Roughly speaking, the first method has $|\mathcal{I}|$ iterations and takes $|\mathcal{I}| c_2 e$; the second has $|\mathcal{T}|^2/2$ iterations and takes $c_1 d |\mathcal{T}|^2/2$; and the third has $d|\mathcal{T}|$ iterations and takes $c_2 e d |\mathcal{T}|$.

There is one problem with the estimations above: we ignored the cost of setting M. For the second and the third approach, they both only need to set each $M_R(x, y)$ once; the first approach however can repeatedly set the same cell. What is even worse, for a large scale dataset, we might be unable to have the full matrix in memory, and thus updating random cells becomes more costly. In contrast, the third approach calculates cells row by row, and both the second and the third approach can stream out the results since each cell is set at most once. When choosing between the second and the third approach, we pick the third one if the ratio $r = \frac{c_1 d|\mathcal{T}|^2/2}{c_2 ed|\mathcal{T}|} = \frac{c_1|\mathcal{T}|}{2c_2 e} > 1$. Note both c_1 and c_2 can be easily estimated by experiment, and c_2 is usually one to two orders of magnitude larger than c_1. In general, if the size of all the tags is small enough to hold the full matrix in memory, then use the first approach; otherwise, if we find in the dataset that each instance usually uses a very small portion of all the tags (e.g. less than 1%), the third approach is preferred than the second. In a multi-source cross-domain dataset such as the BTC dataset, instances usually have very few tags from other domains, e.g. a musician instance will seldom use tags from domains like e-Government or life sciences; thus we use third approach.

This matrix provides a function for each tag t_x to return all the tags that co-occur with it in at least one instance. i.e. $\mathrm{CO}_R(t_x) = \{t_y | M_R(x, y) = 1\}$. We shall discuss the significance of this function next.

4 Online Computation

Given a context T and entailment regime R, the online computation will return all the $f_R(\{t\} \cup T)$ for every tag t. With our index, we can simply issue an IR query for each t that counts all the instances with all tags in T and t, which is getting the number of total hits for a boolean `AND` query. Note that the underlying system compares the posting lists of all tags in the query, and because T is the common part among this series of queries, the intersected posting list can be shared among queries. Thus increasing $|T|$, i.e., the size of the context, may simplify the queries by generating a shorter posting list for T. A quality IR system can answer a count query within a few milliseconds, but since we have hundreds of thousands of tags, we need to focus on how to reduce the number of queries.

There are two special cases of the f_R results: (1) $f_R(\{t\} \cup T) = f_R(T)$; and (2) $f_R(\{t\} \cup T) = 0$. The question is whether we can know the results without issuing f_R queries. For the first case, if t is a super tag of any tag in T, i.e. $\exists t' \in T, t \in \mathrm{super}_R(t')$, adding t to T does not change the instance set and thus does not change f_R. For the second case, ideally we want to skip every tag with an empty intersection without issuing a query. For convenience, we define $Z_R(T) = \{z | f_R(\{z\} \cup T) = 0\}$. Let CL be the candidate list of tags whose queries are finally issued. We propose three different pruning approaches to make CL as short as possible.

1. **Use the Co-occurrence Matrix (M).** Given T, $\bigcap_{t' \in T} \mathrm{CO}_R(t')$ has (and not necessarily only has) all the tags $\{t | f_R(\{t\} \cup T) > 0\}$. When $|T| = 1$, it returns only the co-occurring tags and prunes all the $Z_R(T)$. When $|T| > 1$, it returns a super set of the co-occurred tags, because the returned tags are only known to pairwise co-occur with any tag in T, but are not guaranteed to co-occur with all tags in T in the same instance.
2. **Use the previous tag cloud cache (C).** Since $\mathrm{Inst}_R(\{t\} \cup T) \subseteq \mathrm{Inst}_R(T)$, the set of co-occurred tags given context $\{t\} \cup T$ is also a subset of that given T. Thus if we cache the previous tag cloud, which has the same context T except for the most recently added tag, we can get another super set of the co-occurred tags for context T. This relies on the tag cloud application scenario: it is very likely that the current request is from a user adding a new tag to the context. However we believe it can be applied to any scenarios involving a depth-first search of the context space.
3. **Dynamic update (D).** When computing $f_R(\{t\} \cup T)$ for all the candidate tags from the above two approaches, if we find $f_R(\{t_x\} \cup T) = 0$, we know $\forall t_y \in sub_R(t_x), t_y \in Z_R(T)$, and these tags will be ignored in further computation. This approach can be optimized if we sort the list of tags such that sub tags always follow super tags. However, our tag cloud system does not use this optimization because it needs to stream results alphabetically.

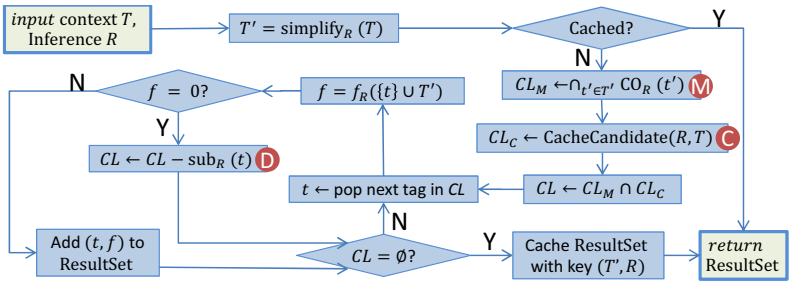


Fig. 3. Pruning for Online Computation

The online computation works as shown in Fig. 3, where the pruning steps are marked with red circles. First, the input context T will be simplified (under R-Inference) to its semantic-equivalent T' so that any redundant tags will be removed (e.g., if $T = \{t_1, t_2\}$ and t_1 is a super tag of t_2 then $T' = \{t_2\}$) and any equivalent tag will be changed deterministically to a representative tag. Then the system checks whether this semantic-equivalent request has been kept in cache for direct output. If not, the system will get candidate lists CL_M from the first approach using T' and CL_C from the second approach using T. Then we use the intersection $CL = CL_M \cap CL_C$ as the candidate list for queries and keep updating it using the third approach. It is easy to prove that using simplified T' in the first approach will get the same candidate tags as using T. Given $T = \{t_1, t_2\}$

where t_1 is a super tag of t_2, we can see $\mathrm{CO}_R(t_1) \cap \mathrm{CO}_R(t_2) = \mathrm{CO}_R(t_2)$ since $\mathrm{CO}_R(t_1) \supseteq \mathrm{CO}_R(t_2)$. However by removing super tags, we can avoid unnecessary intersection of lists when computing the candidates. On the other hand, the cache approach needs the original T in order to get the previous context; subsequently, this previous context is simplified for cache lookup.

5 Supporting Different Entailment Regimes

From the raw dataset, we get only $\mathrm{Tags}_\emptyset$, the tags of each instance with no inference applied. In order to implement Tags_R, Inst_R and CO_R for different R, we can either materialize them so that they serve as independent repositories; or we can always do the inference on-the-fly. We first discuss how to represent the three functions under R by combining the $R = \emptyset$ versions (i.e., with no inference) with the tag subsumption functions super_R and sub_R. After that we will discuss the design choice regarding materialization.

By adding inference, an instance will be assigned with the super tags of its explicit tags, and a tag will be assigned to all instances of its sub tags. i.e.

$$\mathrm{Tags}_R(i) = \bigcup_{t' \in \mathrm{Tags}_\emptyset(i)} \mathrm{super}_R(t') \tag{1}$$

$$\mathrm{Inst}_R(T) = \bigcap_{t \in T} \bigcup_{t' \in \mathrm{sub}_R(t)} \mathrm{Inst}_\emptyset(t') \tag{2}$$

From Eq. (1), we know that

$$t \in \mathrm{Tags}_R(i) \Leftrightarrow \exists t' \in \mathrm{sub}_R(t), t' \in \mathrm{Tags}_\emptyset(i) \tag{3}$$

If tag s co-occurs with tag t under R,

$$\begin{aligned} & s \in \mathrm{CO}_R(t) \Leftrightarrow \exists i \in \mathcal{I}, s \in \mathrm{Tags}_R(i) \wedge t \in \mathrm{Tags}_R(i) \\ \Leftrightarrow\ & \exists i \in \mathcal{I}, \exists s_x \in \mathrm{sub}_R(s), \exists t_y \in \mathrm{sub}_R(t), s_x \in \mathrm{Tags}_\emptyset(i) \wedge t_y \in \mathrm{Tags}_\emptyset(i) \\ \Leftrightarrow\ & \exists s_x \in \mathrm{sub}_R(s), \exists t_y \in \mathrm{sub}_R(t), s_x \in \mathrm{CO}_\emptyset(t_y) \end{aligned} \tag{4}$$

For convenience, we define

$$\mathrm{super}_R^\cup(T) = \bigcup_{t' \in T} \mathrm{super}_R(t') = \{t | \exists t' \in T, t \in \mathrm{super}_R(t')\} \tag{5}$$

And similarly,

$$\mathrm{CO}_R^\cup(T) = \bigcup_{t' \in T} \mathrm{CO}_R(t') = \{t | \exists t' \in T, t \in \mathrm{CO}_R(t')\} \tag{6}$$

Then we compute $\mathrm{CO}_R(t)$ from Eq. (4).

$$\begin{aligned} \mathrm{CO}_R(t) &= \{s | \exists s_x \in \mathrm{sub}_R(s), \exists t_y \in \mathrm{sub}_R(t), s_x \in \mathrm{CO}_\emptyset(t_y)\} \\ &= \{s | \exists t_y \in \mathrm{sub}_R(t), \exists s_x \in \mathrm{CO}_\emptyset(t_y), s \in \mathrm{super}_R(s_x)\} \\ &= \mathrm{super}_R^\cup(\{s_x | \exists t_y \in \mathrm{sub}_R(t), s_x \in \mathrm{CO}_\emptyset(t_y)\}) \\ &= \mathrm{super}_R^\cup(\mathrm{CO}_\emptyset^\cup(\mathrm{sub}_R(t))) \end{aligned} \tag{7}$$

In our implementation, as shown in Fig. 2, we materialize Tags_R for all 4 entailment regimes, thus we do not need to compute Eq. (2) for online computation. However we only precompute $\text{CO}_\emptyset$ and use Eq. (7) at online computation. We made our design choices based on two reasons. First, How much slower will it be if not materialized? Both Eq. (2) and (7) include union and intersection of sets or posting lists, however the lists of instances are usually much larger and using Eq. (2) significantly increases the execution time compared to the materialized index. Second, How important is the runtime performance? As in our scenario, for each tag cloud (or conditional distribution) given T, CO_R is only called once, however Inst_R is called for each tag from the candidate set.

Also note Eq. (7) can be used for either online computation of CO_R or precomputation if it is materialized. Building the co-occurrence matrix M_R is a time consuming step (see Fig. 4). We should avoid repeating it four times for four inference regimes. Instead, we only need to build $M_\emptyset$, which is the easiest because each instance has the minimal number of tags, and the co-occurrence for all the other inference regimes can be computed based on Eq. (7).

6 Experiments

Our system is implemented in Java and we conducted all experiments on a RedHat machine with a 12-core Intel 2.8 GHz processor and 40 GB memory.

In order to test the performance of our preprocessing approach, we apply it to all five subsets of the BTC 2012 dataset, as well as the full dataset. The statistics are listed in Table 1.

Table 1. Statistics of Triples in the subsets of BTC 2012 dataset

Set Name	Total	Ontology Triples	SameAs Triples	Instance Triples
`rest`	22 M	54.7 K	734 K	~22 M
`freebase`	101 M	0	897 K	92 M
`dbpedia`	198 M	1.8 K	22,818 K	175 M
`timbl`	205 M	1,260.1 K	340 K	203 M
`datahub`	910 M	466.0 K	4,490 K	905 M
full set	1,437 M	1,782.6 K	37,357 K	1,397 M

Fig. 4 illustrates how long each step of preprocessing takes for each subset. The Multi-Inference step is not included in the figure since it is too short (41s for the full set) compared with other steps. In general the sorting step and the steps that involve a full scan of the dataset, such as Replace/Flip and index, are the most substantial. Each step is related to certain factors of the dataset provided in Table 1. E.g. the time for inference is related to the number of tag subsumption axioms, which is correlated with the number of ontology triples; the time for union-find on sameAs is related to the number of SameAs triples; and most of the other steps are related to the number of instance triples. Despite the

differences in the portions of different kinds of triples, we also plot the time/space for datasets against their numbers of total triples in Fig. 5, which shows the scalability of our preprocessing approach. The reported disk space includes both the index and the no-inference co-occurrence matrix ($M_\emptyset$), and is dominated by the index, which usually takes $> 90\%$. We can see the time is quite linear with the total number of triples, because most of the major steps are linear w.r.t. the number of triples. The space however is slightly less correlated to the total number of triples, since many different triples might only contribute to a single tag in the index. For example, there might be 1000 triples saying a `foaf:Person` `foaf:knows` 1000 different people, however these triples only contribute a single property tag to this person. This is exactly what happens in the `timbl` subset, and explains why we see `timbl` has slightly more triples than `dbpedia` but needs less time/space.

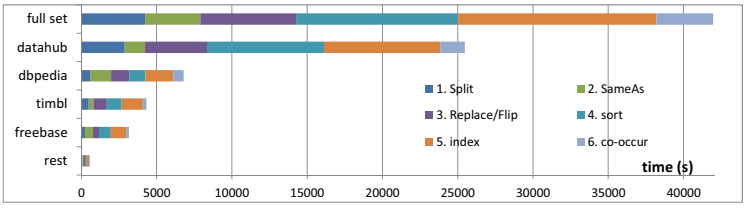


Fig. 4. Time for steps of preprocessing various datasets

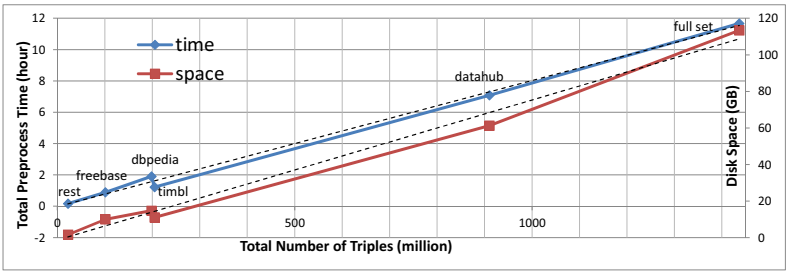


Fig. 5. Preprocessing: Time/Space - Total Triples

We then test the response time of $f_R(\{t\} \cup T)$ queries, i.e. how long it takes to count the instances of tag t with context T by querying the index. To ensure a random but meaningful context T, i.e. $\text{Inst}_R(T) \neq \emptyset$, we randomly pick an instance i and get a subset (size of 6) from its tags $\text{Tags}_\emptyset(i)$ as $[t_{i,1}, t_{i,2}, \ldots, t_{i,6}]$. Thus the six tags in this array are known to co-occur under all entailment regimes. We generate 100 such arrays using different i. Additionally, we pick a set S of 10000 random tags. Starting from[5] $k = 1 \ldots 6$, we use the first k tags

[5] The initial tag cloud ($|T|$=0) is precomputed and cached, thus we do not test it here.

in the arrays as contexts T, and we measure the average time of $f_R(\{s\} \cup T)$ for all $s \in S$. While S might overlap with some T, it does not impact the query time since we issued the same f_R queries without removing redundant query terms. By doing this, we can compare the average query time for different contexts T because they are intersected with the same tags; and we can compare the difference when adding more tags to contexts because as k increases, each array will provide a more "strict" context then before. We also change $R = \emptyset, R_{Sub}, R_{DR}, R_{Sub} \cup R_{DR}$ to examine the impact of different inference. The average time per 10K queries grouped by $|T|$ is shown in Fig. 6. In average, it takes 0.6∼0.7 milliseconds for a single f_R query. The time slightly increases (sub-linear) when we add more tags to context. It takes longer if R has more inference rules due to longer posting lists of tags in the index. As we expect, since there are fewer tags added to each instance from domain/range inference, we find the curves for R_{DR} and $\emptyset$ are close, while R_{Sub} and $R_{Sub} \cup R_{DR}$ are nearly identical.

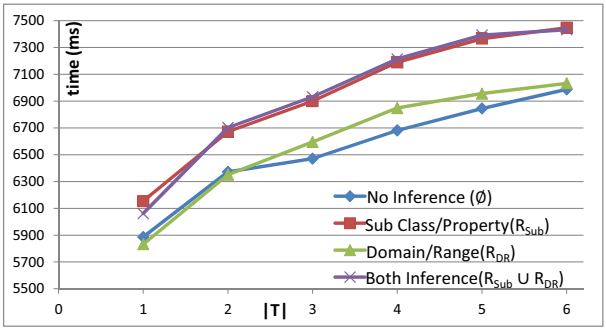


Fig. 6. Average time for 10K queries as context T grows for each entailment regime

A reasonable question is whether a high-performance triple store could be used as a backend for our system. To answer this question, we compare the response time of this specific kind of queries with RDF-3X [10] a state-of-the-art SPARQL engine that "outperforms its previously best systems by a large margin". It takes 9 hours and 11 minutes to load the full BTC dataset into RDF-3X. Note that this loading does not include any kind of inference, sameAs closure/replacement, nor co-occurrence computation as we do in our preprocessing. Similar to the previous experiment, for context size $|T| = 1, \ldots, 5$, we randomly pick 50 (10 of each) contexts, and this time we measure how long it takes for both systems to compute the full contextual tag cloud without pruning. i.e. for a given T, we compute $f_\emptyset(\{t\} \cup T)$ for $\forall t \in \mathcal{T}$. We use $R = \emptyset$ because RDF-3X does not explicitly do any inference. The comparison results are shown in Table 2. In addition to the average execution time of both systems, we also list the Average/Maximum/Minimum Differences, which shows how much faster our system is compared to RDF-3X, with respect to an average query, its best query and its worst query. Note, the

times in this table are longer than those in Fig. 6, because we are issuing ∼380K queries as opposed to 10K. It is clear that our system always outperforms RDF-3X. Averaging across all queries in our test set, our system is 10 times faster than RDF-3X. The differences are more pronounced when $|T|$ increases, although both systems have a sub-linear increase in query execution time as $|T|$ increases. There are two outliers of the Max/Min trends. When $|T| = 1$, the Max Diff. occurs when $f_\emptyset(T) = 49,584,018$, which is the largest set of instances specified by the context in our test set. When $|T| = 5$, the Min Diff. occurs when $f_\emptyset(T) = 143$, which is the smallest set of instances specified by the context in our test set. It is possible that the smaller sizes of instances specified by the context lead to more efficient joins in RDF-3X, allowing it to approach our system's performance. The key point to recognize here is that one-size-fits-all triple stores are not always the best solution for scalable applications. By choosing a carefully constrained user interaction method, we are able to design a specialized infrastructure that can meet our performance requirements. That said, we posit that the systems capable of performing voluminous tag intersections can be used not just for supporting user interfaces, but for data mining and anomaly detection as well.

Table 2. Comparison on Time Cost for Computing Full Tag Cloud (No Pruning)

$\|T\|$	Avg. Time Ours	Avg. Time RDF-3X	Avg. Diff.	Max Diff.	Min Diff.
1	65.8 s	887.6 s	13.5 X	93.2 X	1.71 X
2	84.9 s	516.7 s	6.09 X	15.6 X	2.87 X
3	90.7 s	721.2 s	7.95 X	20.6 X	4.56 X
4	92.8 s	1030.8 s	11.1 X	30.8 X	6.24 X
5	110.3 s	1359.7 s	12.3 X	33.4 X	4.44 X
All	88.9 s	903.2 s	10.2 X	93.2 X	1.71 X

We also test how well our system does for pruning candidate tags under the most complex inference $R = R_{Sub} \cup R_{DR}$. Using the approach above, we generate 100 arrays of length 6 from $\text{Tags}_R(i)$, by changing the length of sub arrays we get 600 random T. As we discussed in the previous section, there are three approaches: by co-occurrence matrix (**M**), by previous cache (**C**), or by dynamic update (**D**). By each combination of approaches, we can count how many f_R queries are finally issued, and see how many queries are pruned. Note there is always some pruning due to super tags of tags in contexts. When using approach C, we always assume the previous cache is available.

The average number of pruned tags is shown in Fig. 7. There are $|\mathcal{T}|$ =389K tags in total however most tags only co-occur with a few other tags. Pruning usually saves us unnecessary queries. We can see when $|T|$ increases any approach will generally prune more tags because more tags in T means a more constrained context. Among the three approaches, M in average prunes more tags, and enabling the other two approaches with M only provides less than 1% more pruning (thus we do not show the overlapping curves for combinations MC, MD and MCD). This justifies the preprocessing for the co-occurrence matrix.

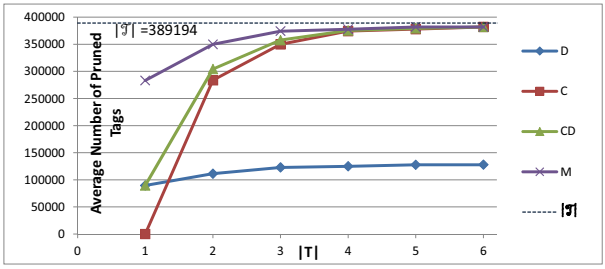


Fig. 7. Average Number of Pruned Tags

C also has good pruning except that when $|T| = 1$, the cache of $|T| = 0$ is a list of every tag and C will not help. However, in the tag cloud scenarios, $|T| = 1$ is important as it will decide the response after the user's first click. Also in practice, the history cache might not always be available (e.g. a user adds t_1, t_2, t_3 and then removes t_2). So its availability is a concern although it requires no preprocessing. The time cost for CO_R is not a key concern to our system. The average time for the above test set is $1.1s$ with all approaches enabled. However running this pruning saves $\sim$300K f_R queries or in average $0.6ms \times 300K = 180s$ for each tag cloud. For the above 600 T, we have an average time of $8.8s$ per tag cloud, with max of $48.8s$. Thanks to the paging and streaming features in our interface design, the first 200 tags in the tag cloud page almost always show within 2 seconds, which we consider an acceptable responsiveness.

7 Related Work

To the best of our knowledge, we have not seen any other works like the contextual tag cloud system, nor papers focusing on optimization for the specific kind of query and resolving related problems. To compare with general purpose triple stores, Rohloff et al. [12] present a comparison of scalability performance of various triple store technologies using the LUBM benchmark [8], and reported that Sesame [4] was the most scalable: It loads around 850M triples in about 12 hours, but it takes more than 5 hours to answer LUBM Query 14, which, similar to our task, requests the instances of a class. Sakr and Al-Naymat [13] survey RDF data stores based on relational databases and classify them into three categories: (1) each triple is stored directly in a three-column table, (2) multiple properties are modeled as n-ary table columns for the same subject, and (3) each property has a binary table. Abadi, et al. [2] explore the trade-off and state the third category is superior to the others on queries. However our previous experiments [17] show using an inverted index is much faster for the queries that count instances of intersections of classes/properties. In this paper we continue to compare our inverted index approach with the state-of-the-art RDF store RDF-3X [10]. The difference in the experiments indicates that a

general purpose SPARQL engine is not always the right choice for a Semantic Web system which requires scalable performance on special kinds of queries.

There are many applications using inverted indices on Semantic Web data. Many of them are Semantic Web search engines. E.g. Sindice [14] and Watson [6] are used to locate Semantic Web documents, while other search engines such as Falcons [5], SIREn [7], and SemSearch [9] are used for locating Semantic entities, and thus whether to index labels, URLs, literal values or other metadata might differ between them. Occasionally, question answering systems [15,16] use inverted indices to help identify entities from natural language inputs, which in some sense is also an entity search engine. Despite the categorization, all the above systems index with keywords because the intended usage is to locate relevant resources based on natural language queries posed by users. Our system is very different because the "terms" in our index are no longer keywords but ontological tags. As a result, our index is compatible with entailments sanctioned by the ontologies in the data. This is also why we propose our preprocessing steps prior to indexing, which we have not seen in other works.

8 Conclusion

The contextual tag cloud system is a novel tool that helps both casual users and data providers explore the BTC 2012 dataset: by treating classes and properties as tags, we can visualize patterns of co-occurrence and get summaries of the instance data. From the common patterns users can better understand the distribution of data in the KB; and from the rare co-occurrences users can either find interesting special facts or errors in the data.

In this paper we discuss the underlying computation problem for the contextual tag cloud system. The main problem we solve is to efficiently compute the conditional distribution of types with respect to the intersection of any number of other types. We use an inverted index for this specific kind of query and propose a scalable preprocessing approach. We also propose pruning approaches to save unnecessary queries. We develop formulas to calculate inference under different entailment regimes. Our experiments verify the scalability of both pre and online computation as well as the effectiveness of our pruning approach.

Although the infrastructure described in this paper is specialized for the contextual tag cloud, we believe this infrastructure can be generally applied to other applications. For example, currently we present the visual patterns to users and rely on human intelligence to recognize any common pattern or unlikely co-occurrence. In the future, we will investigate automated algorithms to learn association rules from or detect anomalies in the dataset.

Acknowledgment. This project was partially sponsored by the U.S. Army Research Office (W911NF-11-C-0215). The content of the information does not necessarily reflect the position or the policy of the government, and no official endorsement should be inferred.

References

1. Abadi, D.J., Marcus, A., Madden, S.R., Hollenbach, K.: Scalable Semantic Web data management using vertical partitioning. In: VLDB, pp. 411–422 (2007)
2. Abadi, D., Marcus, A., Madden, S., Hollenbach, K.: SW-Store: a vertically partitioned DBMS for Semantic Web data management. The VLDB Journal 18(2), 385–406 (2009)
3. Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., Ives, Z.G.: DBpedia: a nucleus for a web of open data. In: Aberer, K., et al. (eds.) ASWC 2007 and ISWC 2007. LNCS, vol. 4825, pp. 722–735. Springer, Heidelberg (2007)
4. Broekstra, J., Kampman, A., van Harmelen, F.: Sesame: A generic architecture for storing and querying RDF and RDF schema. In: Horrocks, I., Hendler, J. (eds.) ISWC 2002. LNCS, vol. 2342, pp. 54–68. Springer, Heidelberg (2002)
5. Cheng, G., Ge, W., Qu, Y.: Falcons: searching and browsing entities on the Semantic Web. In: WWW, pp. 1101–1102 (2008)
6. d'Aquin, M., Motta, E.: Watson, more than a Semantic Web search engine. Semantic Web Journal 2(1), 55–63 (2011)
7. Delbru, R., Campinas, S., Tummarello, G.: Searching web data: an entity retrieval and high-performance indexing model. Journal of Web Semantics 10 (2012)
8. Guo, Y., Pan, Z., Heflin, J.: LUBM: A benchmark for OWL knowledge base systems. Journal of Web Semantics 3(2), 158–182 (2005)
9. Lei, Y., Uren, V.S., Motta, E.: Semsearch: A search engine for the Semantic Web. In: Staab, S., Svátek, V. (eds.) EKAW 2006. LNCS (LNAI), vol. 4248, pp. 238–245. Springer, Heidelberg (2006)
10. Neumann, T., Weikum, G.: The RDF-3X engine for scalable management of RDF data. The VLDB Journal 19(1), 91–113 (2010)
11. Pan, Z., Heflin, J.: DLDB: Extending relational databases to support Semantic Web queries. In: Workshop on Practical and Scaleable Semantic Web Systems, pp. 109–113 (2003)
12. Rohloff, K., Dean, M., Emmons, I., Ryder, D., Sumner, J.: An evaluation of triple-store technologies for large data stores. In: Meersman, R., Tari, Z. (eds.) OTM-WS 2007, Part II. LNCS, vol. 4806, pp. 1105–1114. Springer, Heidelberg (2007)
13. Sakr, S., Al-Naymat, G.: Relational processing of RDF queries: a survey. ACM SIGMOD Record 38(4), 23–28 (2010)
14. Tummarello, G., Delbru, R., Oren, E.: Sindice.com: Weaving the open linked data. In: Aberer, K., et al. (eds.) ASWC 2007 and ISWC 2007. LNCS, vol. 4825, pp. 552–565. Springer, Heidelberg (2007)
15. Unger, C., Bühmann, L., Lehmann, J., Ngonga Ngomo, A.C., Gerber, D., Cimiano, P.: Template-based question answering over RDF data. In: WWW, pp. 639–648 (2012)
16. Walter, S., Unger, C., Cimiano, P., Bär, D.: Evaluation of a layered approach to question answering over linked data. In: Cudré-Mauroux, P., et al. (eds.) ISWC 2012, Part II. LNCS, vol. 7650, pp. 362–374. Springer, Heidelberg (2012)
17. Zhang, X., Heflin, J.: Using tag clouds to quickly discover patterns in linked data sets. In: Workshop on Consuming Linked Data (2011)
18. Zhang, X., Song, D., Priya, S., Heflin, J.: Infrastructure for efficient exploration of large scale linked data via contextual tag clouds. Tech. Rep. LU-CSE-13-002, Department of Computer Science and Engineering, Lehigh University

Statistical Knowledge Patterns: Identifying Synonymous Relations in Large Linked Datasets

Ziqi Zhang[1], Anna Lisa Gentile[1], Eva Blomqvist[2], Isabelle Augenstein[1], and Fabio Ciravegna[1]

[1] Department of Computer Science, University of Sheffield, UK
[2] Department of Computer and Information Science, Linköping University, Sweden
{z.zhang,a.l.gentile,i.augenstein,f.ciravegna}@dcs.shef.ac.uk, eva.blomqvist@liu.se

Abstract. The Web of Data is a rich common resource with billions of triples available in thousands of datasets and individual Web documents created by both expert and non-expert ontologists. A common problem is the imprecision in the use of vocabularies: annotators can misunderstand the semantics of a class or property or may not be able to find the right objects to annotate with. This decreases the quality of data and may eventually hamper its usability over large scale. This paper describes Statistical Knowledge Patterns (SKP) as a means to address this issue. SKPs encapsulate key information about ontology classes, including synonymous properties in (and across) datasets, and are automatically generated based on statistical data analysis. SKPs can be effectively used to automatically normalise data, and hence increase recall in querying. Both pattern extraction and pattern usage are completely automated. The main benefits of SKPs are that: (1) their structure allows for both accurate query expansion and restriction; (2) they are context dependent, hence they describe the usage and meaning of properties in the context of a particular class; and (3) they can be generated offline, hence the equivalence among relations can be used efficiently at run time.

1 Introduction

The Web of Data is a rich common resource with billions of triples available in thousands of datasets and Web documents (including RDFa and microdata annotations) created by a growing number of people, including non expert ontologists (e.g. Web managers generating schema.org microdata annotations). This, however, brings the risk of imprecision in the use of vocabularies (schemas and ontologies), including their systematic misuse. Annotators can misunderstand the semantics of a concept or relation or may not be able to find the right classes and properties to annotate with[1]. This decreases the quality of data and may eventually hamper its usability over large scale. The problem gets even more complex when using several datasets together, which may refer to the same classes, e.g. DBpedia types, but use different sets of properties.

[1] Throughout this paper we use the terms "concept" and "class" interchangeably, to mean a concept defined in a vocabulary (i.e., ontology). Similarly, we use the term "relation" as a synonym for "property".

H. Alani et al. (Eds.): ISWC 2013, Part I, LNCS 8218, pp. 703–719, 2013.

In this paper we focus on a problem found in numerous datasets: the use of classes and properties which are alien to the reference vocabulary provided for the dataset, but that may be equivalent to existing classes or properties in the reference vocabulary, or may be used to extend that vocabulary at data creation time. This issue generates low recall when querying datasets, as the user must guess which properties are actually used in the dataset as opposed to the ones formally defined in the vocabulary. A similar issue also arises when attempting to query interlinked datasets, but only being aware of the vocabulary used in one of them. The linked datasets may use different properties, but still contain overlapping and complementary data. Without exploring property usage in all those datasets, queries may miss relevant parts of the data.

We propose the definition and use of Statistical Knowledge Patterns (SKP) as a mean to address these issues. An SKP is class-specific and encapsulates key information about an ontology class, including synonymous properties used within (and potentially between) datasets. SKPs aim at reducing the complexity of understanding and querying data, by reducing the variety of properties to only include the core properties of the main SKP class, and their characteristics. We propose an unsupervised approach to generate SKPs based on statistical data analysis, and introduce a measure of "synonymity" of two properties of a class, which is used to cluster synonymous properties. Effectively, an SKP addresses the vocabulary heterogeneity of classes based on their usage data, within datasets, or between datasets that are linked through that class. One possible usage of an SKP is query expansion when querying the data underlying the SKP (shown in Sect. 5).

Both pattern extraction and pattern usage are completely automated. The main benefits of SKPs are: that (1) their structure allows for both accurate query expansion and restriction; (2) they are context dependent, hence they can describe the usage and meaning of properties in the context of a particular class and even within a specific dataset (or group of datasets), hence also accounting for synonymity that hold only in specific repositories, domains or communities; and (3) they can be generated offline, hence the synonymity among properties can be used efficiently at run time.

The paper is organised as follows: Sect. 2 describes related work; Sect. 3 introduces the SKP generation method, and Sect. 4 presents the synonymity measure and property clustering in detail; Sect. 5 describes our experiments and discusses the evaluation of our approach; Sect. 6 concludes the paper and discusses future work.

2 Related Work

Knowledge Patterns (KP) have been defined as general templates or structures used to organise knowledge [8]. In the Semantic Web scenario they are used both for constructing ontologies [3,7,14] and for using and exploring them [2,10,11,13].

In the area of ontology engineering several kinds of patterns [7] have been used. On such type is the *Content Ontology Design Patterns* (CODPs), which are small, reusable pieces of ontologies that consist of just a few classes. They represent core concepts in an ontology and are either extracted or re-engineered from ontologies or other data structures. CODPs are similar to SKPs in the way that they also represent concepts with their most distinguishing characteristics. Unlike SKPs however, they have to be created

manually or semi-automatically, and since they are abstract patterns intended for being used as "templates" in ontology engineering they usually lack any direct connection to data and cannot directly (without manual specialisation) be used for querying Linked Data. Since CODPs represent an abstract top-down view, they additionally do not consider aspects such as diversity and synonymy among properties, which is one of the things we focus on in this paper.

The approach closest to ours is the generation of *Encyclopedic Knowledge Patterns* (EKPs) [11], which have been built mainly for usage in exploratory search [10]. The EKP generation process exploits statistics of links from Wikipedia to select which classes are the most representative for describing each concept. The assumption is that if entities of a class A frequently link to entities of class B, then class B is an important descriptor for class A. This information is formalised as small OWL ontologies (the EKPs), each having one main class and relations to other (significantly frequent) classes. The main purpose of EKPs is to filter out irrelevant data when presenting DBpedia entities, while the ability to query for data is not a primary concern. Hence, EKPs mainly contain abstractions of properties, such as "linksToClassB", which expresses the fact that instances of class A commonly link to instances of class B (links which could in many cases in turn be represented by DBpedia properties, but not necessarily). This is however not sufficient for our case, since our main goal is to use SKPs to query actual data. Hence, we propose an extension of EKPs, which also include a sufficient coverage of actual properties of the datasets. Basse et al. [2] also exploit statistics from a specific dataset to produce topic frames of that dataset. In contrast to Nuzzolese et al. [11] they don't produce a pattern for each class but rather generate clusters of classes (up to 15 classes each) that reflect main topics of the dataset, which is again not sufficient for our goal. Presutti et al. [13] explore the challenges of capturing KPs in a scenario where explicit knowledge of datasets is neither sufficient nor straight-forward. They propose a dataset analysis approach to capture KPs and support datasets querying. Our SKPs expand on this work as not only do we capture direct statistical information from the underlying datasets, but also further characterise relevant properties with additional features (e.g. synonymous properties and range axioms), which we show to be beneficial for querying datasets. As well as exploratory purposes, another common usage of KPs is within Query Expansion (QE), and Question Answering (QA) in general. Typical approaches [5] use the lexicalizations of concepts to map natural language to URIs (with NLP techniques) but they may fail to capture synonymous relations with completely different lexicalizations.

A core component of our method of creating SKPs is measuring synonymity between ontology properties. This is related to the work on linking ontological resources in general [6,9,15]. A large amount of work in this area addresses linking ontology classes and data instances, linking properties, however is insufficiently addressed. Typical approaches employ similarity metrics such as string edit distance and semantic similarity measures. However, string similarity fails to identify equivalent relations if their lexicalisations are wholly distinct, which is very common in Linked Data. Semantic similarity often depends on taxonomic structures in existing ontologies [4]. Unfortunately, many relations which are used in Linked datasets are invented arbitrarily or originated from rudimentary ontologies [12]. Our previous research [1] shows that a bottom-up

approach that uses Linked Data statistics offers effective solution to measuring similarity. Therefore in this work we introduce a data-driven synonymity measure for properties on Linked Data and we use it in the construction of SKPs.

3 SKP Construction Overview

A Statistical Knowledge Pattern (SKP) is an ontological view over a class (defined in a reference ontology), and captures and summarises the usage of that class (hereafter called the *main class* of the SKP) in data. An SKP is represented and stored as an OWL ontology. The term "statistical" refers to that the pattern is constructed based on statistical measures on data. Each SKP contains: (1) properties and axioms involving the main class derived from a reference ontology; (2) properties and axioms involving the main class that are not expressed in the reference ontology, but which can be induced from statistical measures on statements published as Linked Data.

The generation of SKPs is mainly characterized by the identification (based on data triples) and selection of (1) synonymous (i.e. interchangeable) properties; (2) ranges for properties that have no prior range in the reference ontology. Not all properties (or clusters of synonymous properties) are stored in the final SKP. To decide which ones are representative of the SKP main class, their *relevance* is measured based on the frequency of usage in available data. The information encoded in the SKP is specific to the main class, i.e., it does not show a general interpretation of the involved properties but rather the specific way they are used with the main class. For example, the same property may be present in several SKPs, but with distinct range axioms and as part of separate property clusters, depending on how it is used with the respective main class of each SKP. A concrete example is the property *dbp:lakeName*[2] which is synonymous to *foaf:name*, but only for the class *dbo:Lake*.

At the moment we only consider the properties that are used with instances of the main class in the subject position, as part of the characterization of that class, which in our experience is usually the case for Linked Data, e.g., the way the DBpedia Ontology is structured. One may also consider properties with the opposite "direction", i.e., instances of the main class as objects (for example, *isLocationOf* instead of *hasLocation*), however, we do not include this at the moment (acknowledging the risk of loosing some fraction of the data) since we have no method to determine what the main focus of a triple is, we therefore make the simple assumption that being a subject of a triple means that this is the entity the triple is describing. The SKP generation is fully automated, whereby SKPs can be re-generated as soon as data change, without manual effort. At the same time SKPs are used as stored resources hence increasing usage efficiency.

Relation to EKP Extraction. Since SKPs are an extension of EKPs [11], if an EKP already exists it can be used as an abstract frame for the concrete properties and axioms

[2] Prefixes used are $dbo :< http://dbpedia.org/ontology/ >$,
$dbp :< http://dbpedia.org/property/ >$,
$skos :< http://www.w3.org/2004/02/skos/core\# >$,
$foaf :< http://xmlns.com/foaf/0.1/ >$,
$rdfs :< http://www.w3.org/2000/01/rdf-schema\# >$

that are added through our SKP generation method. In particular, the abstract properties introduced by EKPs (i.e., "links to class X") can be used to group properties with overlapping range axioms, to give the SKP a more intuitive structure and improve human understandability of the pattern. However, we do not restrict an SKP to being an EKP, i.e., a set of "paths" in DBpedia (c.f. Def. 2 in [11]), but rather the SKP notion is both independent of what reference ontology and datasets are used, and the resulting SKPs may include any OWL axioms that can be statistically induced from data. Our method for extracting the patterns also differs significantly from the EKP extraction method in [11], i.e., we use triple data from the dataset in focus (DBpedia is used as an example for the experiments) while the EKPs are extracted from a wikilink dataset, which means that we operate on triples using distinct named properties rather than just "links". With respect to comparing the methods, we are not using the EKP notion of "path", c.f. [11] where a path is defined as a triple with the first element being the subject type, the second being the property *dbo:wikiPageWikiLink* and the third being the object type, i.e., the first and last element of a path being classes. In our case we are not initially interested in the object types, but rather the object instances (resources) themselves, hence, the path abstraction is replaced by actual RDF triples, with concrete instances (resources) as subjects and objects, but selected based on the fact that the subject type is the main class of the SKP. This means that, for instance, we also include datatype properties, and triples where the object type is missing, as opposed to the EKP extraction method. Similarly to [11], however, we only use single triples, not chains of triples. Paths are selected for EKP inclusion based on their so called "path popularity", i.e., a measure on how large fraction of the individuals of the main class have links to an individual of a certain type (c.f. indicators in [11]), putting the focus on the object types (classes). SKPs, on the contrary, put the main focus on the properties, and apply a frequency measure on property usage, i.e. subject-object pairs of RDF triples using that property (c.f. experiments on different such measures and thresholds in Sect. 5.1), where subjects are all instances of the SKP main class, for setting an inclusion threshold.

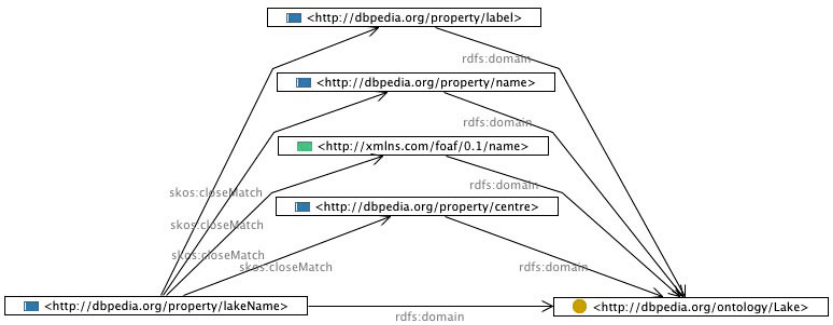


Fig. 1. Extract from the SKP for the DBpedia Ontology class Lake (in TopBraid Composer's RDF graph notation)

SKP Example. As an example, we take the SKP generated for the DBpedia Ontology class *dbo:Lake*[3]. A property from the reference ontology, which is sufficiently frequent in actual data to be included in the SKP, is *dbo:shoreLength*, hence, it is included in the SKP. Additionally, it has been found to be synonymous to the property *dbp:shore*, hence a *skos:closeMatch* assertion is added for the two properties. In this case r_i = *dbo:shoreLength* is a reference property having a set of "synonymous properties" SR_i induced from data (in this case consisting of only one member). Figure 1 illustrates an extract from the same SKP showing another example; the property *dbp:lakeName* has been clustered with the properties *dbp:name*, *dbp:label*, *foaf:name*, and *dbp:centre*, which is expressed via the *skos:closeMatch* assertion. The selection of synonymous properties is obtained via a *synonymity* measure, described in Section 4, but before taking a deeper look at the measure we provide an overview of the steps of the two main phases of the overall method; detecting synonymity, and selecting properties.

Synonymity of Properties. To create an SKP we identify the properties used for the SKP main class based on data and measure their synonymity. We propose a novel synonymity measure of properties or relations to be detailed in Section 4. The overall process is:

1. Query the dataset for all the instances (IND) of the main class; query the dataset for all triples having any $i \in IND$ in subject position (IND_{subj}) and collect the types (through *rdf:type* or a datatype) of the objects of all those triples.
2. For each property used in IND_{subj}, collect the subset of IND having the property as predicate, IND_{prop} - the *subject-object* pairs of this set represents the characteristics of that property, given the main class at hand.
3. Do a pairwise comparison of all subject-object pairs in IND_{prop} for all the properties and calculate a *synonymity* score for each pair.
4. Use the *synonymity* scores (representing evidence of properties being interchangeable) to cluster properties representing the same semantic relation.

Selection of Properties. The aim of the above process is to discover for each specific main class clusters of properties with the same meaning. In practice, certain number of properties are found to be noise or non-representative. Thus we further refine the set of selected properties for each SKP as follows:

1. Calculate the frequencies of properties used in data, i.e. counting distinct objects in IND_{prop}. For clusters, treat the cluster as if it was a single property hence add the frequency counts of the constituent properties.
2. Use a cutoff threshold T to filter out unfrequent properties (or clusters). Add those above the threshold to the SKP, including information about their appropriate property type (e.g. *owl:DatatypeProperty* or *owl:ObjectProperty*), with their original namespace intact. We experiment with several approaches for setting the threshold T, and report on this in Section 5.
3. For each member of a property cluster that is added to the SKP, add a *skos:closeMatch* relation between the cluster members.

[3] `http://ontologydesignpatterns.org/skp/Lake.owl`

4. For each property, create the set of possible ranges. Construct a range axiom including each range class that is given to the property in the reference ontology (if present), and if no range axiom is present, construct a range axiom as the union of each range class that can be identified in data (frequent object types of the triples).
5. Add *rdfs:subPropertyOf* axioms for those properties where the ranges match some abstract EKP property (i.e., the "links to class X" abstract properties).
6. Store the SKP as an OWL2 file.

4 Synonymity Measure and Property Clustering

We consider synonymity to be symmetric and argue that the synonymity for each distinct pair of properties or relations depends on three components: triple overlap, cardinality ratio and clustering.

Triple Overlap. evaluates the degree of overlap in terms of the usage of properties in triples. Let p be a property and r_p be the set of triples containing p as predicate, and let $SO(p)$ be the collection of subject-object pairs from r_p and SO_{int} the intersection

$$SO_{int}(p, p') = SO(r_p) \cap SO(r_{p'}) \tag{1}$$

then the triple overlap $TO(p, p')$ is calculated as

$$MAX\{\frac{|SO_{int}(r_p, r_{p'})|}{|r_p|}, \frac{|SO_{int}(r_p, r_{p'})|}{|r_{p'}|}\} \tag{2}$$

Intuitively, if two properties p and p' have a large overlap of subject-object pairs in their data instances, they are likely to have identical meaning. The *MAX* function minimises the impact of infrequently used, but still synonymous relations (i.e., where the overlap covers most triples of an infrequently used relation but only a very small proportion of a much more frequently used).

Subject Agreement. While triple overlap looks at the data in general, subject agreement looks at the overlap of subjects of two relations, and the degree to which these subjects have overlapping objects. Let $S(p)$ return the set of subjects of relation p, and $O(p|s)$ returns the set of objects of relation p whose subjects are s, i.e.:

$$O(p|s) = O(r_p|s) = \{t_o | t_p = p, t_s = s\} \tag{3}$$

we define:

$$S_{int}(p, p') = S(r_p) \cap S(r_{p'}) \tag{4}$$

$$\alpha = \frac{\sum_{S \in S_{int}(p,p')} \begin{cases} 1 & \text{if } |O(p|s) \cap O(p'|s)| > 0 \\ 0 & \text{otherwise} \end{cases}}{|S_{int(p,p')}|} \tag{5}$$

$$\beta = \sqrt{|S_{int}(p, p')|/|S(p) \cup S(p')|} \tag{6}$$

then the agreement $AG(p, p')$ is

$$AG(p, p') = \alpha \cdot \beta \tag{7}$$

In Equation 7, α counts the number of overlapping subjects whose objects have at least one overlap. The higher the value of α, the more the two relations agree in terms of their shared subjects. We do not consider the absolute value of overlap because both p and p' can be 1:many relations and a low overlap value could mean that one is densely populated while the other is not, which does not necessarily mean they do not agree. β evaluates the degree to which two relations share the same set of subjects. The agreement $AG(p, p')$ balances the two factors by taking their product. As a result, relations that have high level of agreement will have more subjects in common (β), and a large proportion of shared subjects who also have shared objects (α).

Cardinality Ratio is a ratio between cardinality of the two relations. Cardinality of a relation $CD(p)$ is calculated based on data:

$$CD(p) = \frac{|r_p|}{|S(r_p)|} \tag{8}$$

and the cardinality ratio is calculated as

$$CDR(p, p') = \frac{MIN\{CD(p), CD(p')\}}{MAX\{CD(p), CD(p')\}} \tag{9}$$

On a sufficiently large sample, the derived cardinality value should be close to the conceptually true value and two equivalent relations should also have the same cardinality. The final synonymity measure integrates all the three components to return a value in [0, 2]:

$$E(p, p') = \frac{TO(p, p') + AG(p, p')}{CDR(p, p')} \tag{10}$$

Clustering. We apply the measure to every pair of relations of a concept of interest, and keep those with a non-zero synonymity score. The goal of clustering is to create groups of synonymous relations based on the pair-wise synonymity scores. We use a simple rule-based agglomerative clustering algorithm that uses a number of thresholds. First, we build initial clusters based on all property pairs. We rank all property pairs by their synonymity score, then we keep a pair as an initial cluster if (i) its score and (ii) the number of triples covered by each property are above a certain threshold, T_{minSyn} and T_{minTP} respectively (i.e., we create a cluster containing p and p' if $E(p, p') > T_{minSyn}$ and $|r_p| > T_{minTP}$ and $|r_{p'}| > T_{minTP}$). Next, to merge clusters, given an existing cluster C and a new pair (p, p') where either $p \in C$ or $p' \in C$, the other property is added to C if $E(p, p')$ is close to the average of all equivalence scores of connected pairs in C. The closeness is determined by another threshold $T_{minSynRel}$, which is a fractional number. Thus if the average of the equivalence scores of all connected pairs in C is 0.7, the new pair (p, p') is merged with C if $E(p, p') > 0.7 * T_{minSynRel}$. This preserves the strong connectivity in a cluster. This is repeated until no further merge action is taken.

5 Evaluation

To evaluate the use of SKPs we have chosen to focus on two main aspects: (1) the extent to which the SKPs describe and characterise the underlying data, and give access to that data; (2) the extent to which the identification of "synonymous" properties improve on retrieval coverage, without introducing erroneous data in the result. Thus we perform two sets of experiments for different purposes.

We used DBpedia as the underlying semantic data repository in this experiment and we query the live DBpedia SPARQL endpoint to retrieve data, and use the DBpedia Ontology as reference ontology[4]. In this evaluation, we created SKPs for 34 DBpedia classes[5], which are exactly the classes that can be extracted from the queries of the QALD1 question answering dataset[6]. These classes were selected for the experiments since we intend to in the future apply the SKPs to query expansion and make use of the QALD1 dataset as a benchmark. The SKP construction method is not restricted to any particular ontology or datasets, but can be applied to any reference ontology-dataset pair for which they are needed. In the experiments we use $T_{minSyn} = 0.1$, $T_{minTP} = 0.01\%$ and $T_{minSynRel} = 0.6$ as thresholds.

5.1 SKP Observation

SKPs aim at reducing the variety of properties to only include the core properties of the main SKP class, however, to be useful in practice, such a reduced representation should still allow for accessing as large part of the underlying data as possible. We have measured two aspects of each SKP, (1) the absolute number of properties included in the SKP and the fraction of the total number of distinct properties of the main class that this set represents, and (2) the fraction of the total number of triples (where the subject is an instance of the main class) that the properties included in the SKP allows to cover. The ideal situation would be that a low absolute number (1) of properties would still render an almost perfect coverage of triples (2).

However, first we need to generate a set of SKPs to assess, and for this we need to set an appropriate property selection threshold. Three sets of experiments have therefore been performed, each applying a different method for setting the threshold on what properties to include in the SKP. For each such threshold, the parameters of the threshold have been varied, so as to evaluate (a) the amount of properties included, and (b) the amount of triples covered, in each case. This leads us to conclude both which method for setting the threshold that seems to perform best over the SKP test set, and also gives us an evaluation of how well the SKPs using that threshold perform on criteria (1) and (2).

A naive approach would be to set an *absolute threshold* on the count of triples using a certain property, i.e. including all properties with more than a certain number of

[4] `http://dbpedia.org/sparql`, ontology: `http://dbpedia.org/ontology`

[5] The preliminary SKPs used in the evaluation, including skos:closeMatch statements, can be found at `http://ontologydesignpatterns.org/skp/SKPs130510.zip`

[6] `http://greententacle.techfak.uni-bielefeld.de/~cunger/qald1/evaluation/dbpedia-test.xml`

triples (subject type being the SKP main class). Figure 2 shows the performance of this approach, illustrating both the best (maximum triple coverage and minimum fraction of included properties), worst (minimum triple coverage and maximum fraction of included properties) and the average performance of each criteria (average triple coverage and average fraction of included properties), over the SKP set. At an absolute threshold of 20 triples we have a triple coverage between 79 and 98%, at an included property fraction of between 14 and 45%.

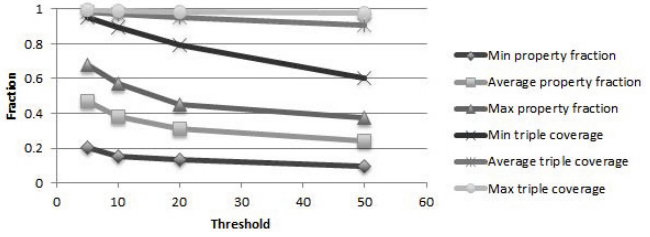


Fig. 2. Performance of the absolute thresholds (when set to 5, 10, 20 and 50 triples)

A more elaborate threshold would consider the properties that represent at least *a certain fraction* of the total number of triples (subject type being the SKP main class). Although this may sound reasonable at first glance, Figure 3 shows that this threshold actually performs worse than the absolute threshold. There is actually no value of the fraction that we could find which both guarantees us to get any properties at all, for all the SKPs in our set, but with no SKP on the other hand including the complete set of properties (even those with very low frequency). As in the previous figure, Figure 3 shows the best, worst, and average performance over the SKP set. While we can get the worst case property fraction included to drop to about 75% (at the 1% threshold) then the coverage of triples has already dropped to below 70%.

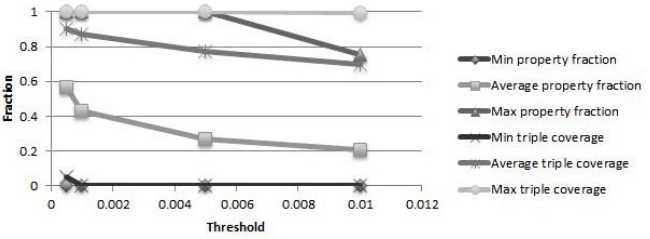


Fig. 3. Performance of the triple fraction threshold (when set to between 0.05% and 1% of triples)

Finally, we explore a *normalised threshold* that turns out to perform best. We first calculate the average number of triples per property (where the property set still constitute all triples where the subject type is the main class), and then set a threshold as a fraction of that average. With this threshold we, hence, both take into account the size

of the triple set (as in the previous method), but also the number of properties used for instances of the main class. In Figure 4 the performance of this threshold is shown. Note that up until around 0.6 (meaning that properties are included if the size of their triple set constitute at least 60% of the average size of a triple set for any property used for the main class) the triple coverage stays really high, i.e. in the worst case still above 88%, with a worst case fraction of included triples of 38%. Hence, this indicates that using this threshold, we can probably discard at least 62% of all properties (usually more), for *any* class in the DBpedia ontology, without reducing the amount of triples we can still access to below 88% (on average we can even access 94% of the triples, or in the best case 97%).

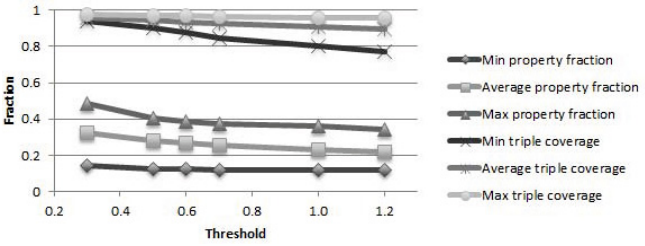


Fig. 4. Performance of the final threshold (when set to between 0.3% and 1.2% of the average number of triples per property)

Although we expect this to be true also for other datasets, and for generating SKPs over several interlinked datasets, we can of course not guarantee that this is the best threshold in all cases. However, this experiment also shows how one can (completely automatically) test a threshold calculation method, to select the best one. Hence, when generating SKPs for other datasets, it is recommended to rerun these experiments, to find the "optimal" threshold for those datasets.

Nevertheless, the 0.6 threshold has been used for the SKPs generated for the rest of our experiments, and Table 1 shows the characteristics of the SKPs generated using this threshold. The name of the SKP is equal to the name of the main class, as defined in the DBpedia Ontology (our reference ontology). First we present some statistics on the main class itself, i.e. the number of instances in DBpedia version 3.8, the total number of triples with those instances in the subject position, and the total number of distinct properties that can be found in that set of triples. Next, we present some statistics on the SKP generated for that main class, i.e. the total number of properties included in the SKP, the fraction of the total number of properties the select ones represent (criteria (1) mentioned previously), and the fraction of the included properties that are not defined in the reference ontology. The latter aspect gives a first indication of how much added information about the main class our SKPs contain, compared to the DBpedia Ontology itself. On average, 78% of the properties in our SKPs are not defined in the DBpedia ontology. Finally, we present the ability of the SKP to cover the actual triples in the DBpedia dataset, for the main class, through the number of triples covered and the fraction of the total number of triples that the selected ones represent (criteria (2)).

5.2 Using SKPs for Query Expansion

To evaluate the benefit of synonymous properties provided by the SKP we create a query expansion experiment to study (i) the increase in recall (added data) with (ii) the decrease in precision (introduced errors) by using the added properties. For each SKP we consider the set R_{ont} of all properties defined by a reference ontology - DBPedia in this case, and we generate a query for each $r_i \in R_{ont}$ such as "SELECT DISTINCT ?s ?o WHERE {?s a Main_Concept_of_SKP . ?s r_i ?o .}". The set of values of ?*o* returned from each query is considered the *baseline value set* for the property r_i, denoted by V_{r_i}. Then let SR_i be the set of synonymous properties to r_i. For each $r_i \in R_{ont}$ we generate a query for each $sr_i^j \in SR_i$ in the same form as before to retrieve the set of values $V_{sr_i^j}$. Then the total *expanded value set* (denoted by EV_{r_i}) is the union of all $V_{sr_i^j}$ for each sr_i^j. To determine the accuracy of values in EV_{r_i}, we manually analyse each property in each SR_i and annotate the property as either a *correct synonymous property* for r_i or not. Annotating property synonymity involved four computer scientists, and the Inter-

Table 1. Characteristics of the generated SKPs

SKP name (main class)	no. of instances	no. of triples	no. of properties	no. of properties in SKP	fract. of properties included	fract. of non-ontology properties	no. of triples covered	fract. of triples covered
Actor	2912	19833	246	88	0.36	0.77	18847	0.95
AdministrativeRegion	28229	20721	1235	436	0.35	0.90	18432	0.89
Agent	956476	18395	877	232	0.26	0.69	16197	0.88
Architect	1348	19540	169	38	0.22	0.74	18786	0.96
Bridge	2775	23446	351	94	0.27	0.68	21711	0.93
BritishRoyalty	6563	17918	250	66	0.26	0.79	17037	0.95
Cave	238	6100	105	31	0.30	0.87	5677	0.93
City	24423	27064	876	238	0.32	0.89	24210	0.89
Company	44516	17010	352	102	0.29	0.63	16037	0.94
Country	2710	22530	666	194	0.29	0.87	21286	0.94
Currency	333	6807	173	64	0.37	0.97	6072	0.89
Eukaryote	199085	15983	255	78	0.31	0.85	14853	0.93
FictionalCharacter	9878	19441	377	120	0.32	0.85	18352	0.94
Film	88503	17674	153	48	0.32	0.65	16847	0.95
Lake	10294	19915	412	51	0.12	0.63	19140	0.96
Language	6860	19357	149	35	0.23	0.89	17942	0.93
Magazine	3388	18986	252	41	0.16	0.78	18427	0.97
MilitaryConflict	10691	20904	187	32	0.17	0.66	20209	0.97
Model	1416	18576	257	63	0.25	0.84	17770	0.96
Mountain	13259	19895	359	65	0.18	0.82	18799	0.94
MountainRange	1691	26717	322	106	0.33	0.79	25089	0.94
Museum	3148	17631	290	44	0.15	0.80	16831	0.95
OfficeHolder	32373	21706	476	103	0.22	0.65	20039	0.92
Organisation	200789	23777	840	194	0.23	0.69	21434	0.90
Person	763644	17479	580	168	0.29	0.67	15649	0.90
Place	638879	25527	1126	280	0.25	0.86	22751	0.89
PoliticalParty	3311	20822	267	84	0.31	0.75	19903	0.96
Protein	12042	16513	178	48	0.27	0.75	16037	0.97
River	24267	18759	333	128	0.38	0.74	17382	0.93
Royalty	6563	17283	237	63	0.27	0.78	16315	0.94
SoccerClub	15727	30454	192	47	0.24	0.81	29485	0.97
Species	202339	16097	265	75	0.28	0.84	14933	0.93
TelevisionShow	23480	24595	292	95	0.33	0.65	23681	0.96
Website	2388	15555	295	48	0.16	0.85	14773	0.95

Annotator-Agreement (IAA) based on a sample is 0.72. Then, all the corresponding values returned by the respective query that uses this property are marked either correct or wrong. The rationale is that we assume each individual triple on the Linked Data to be semantically correct, or in other words, that data publishers have respected the semantic meanings of predicates when releasing triple datasets. Although this is not always true on the Linked Data, we believe this gives a reasonable approximation of accuracy.

Using these annotations, we study the accuracy of the synonymity measure and also the increase ratio in the retrievable data due to the inclusion of each synonymous property. We create two sets of statistics for this purpose. First, given a reference property r_i and each of its synonymous property sr_i^j, we compute an *increase ratio* as $IR = \sqrt{\frac{|V_{sr_i^j}|}{|V_{r_i}|}}$. We rank all pairs of $\langle r_i, sr_i^j \rangle$ by descending order of their synonymity scores, and plot IR for each pair (Figure 5). In total there are 561 pairs of relations for all SKPs.

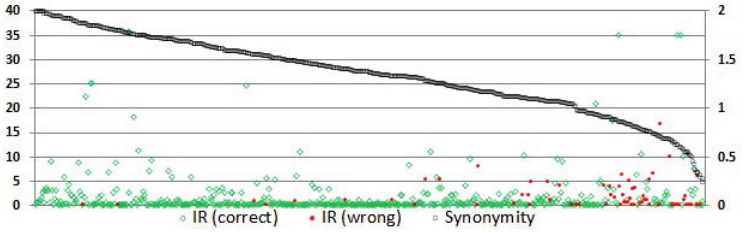


Fig. 5. Increase Ratio (IR) for each $\langle r_i, sr_i^j \rangle$ (marked as green $\diamond$ where the synonymous property is correct and red $\times$ where it is wrong), ranked by synonymity scores ($\square$) in descending order. IR is aligned to the left y-axis and synonymity scores are aligned to the right.

As shown in Figure 5, the synonymity measure correctly predicts synonymous properties in most cases. In many cases, the increase ratio is significant, suggesting that data retrieval can considerably boost recall by including SKP synonymous properties in the query process. There appears to be an inverse correlation between the synonymity score and the correctness of prediction. With high synonymity scores (e.g., > 1.4) the vast majority of synonymous properties discovered for the reference properties are found correct; however, when errors are made, the increase ratio is very low, meaning little noise is added. This is a useful feature as, when necessary, we can apply higher threshold in order to ensure high precision in retrieval. There is no correlation between the increase ratio and the synonymity score. This is expected as on the one hand, synonymity describes the extent to which two properties are "interchangeable" while the increase ratio addresses what they have "in difference". Ideally, for the purpose of data retrieval, we would like to have a re-ranking process to combine both synonymity and increase ratio in order to promote properties that are highly synonymous, and can also potentially add a lot information to each other. We will explore this in future.

While Figure 5 looks at individual pairs of properties independently, from the SKP construction point of view we are more interested in the incremental performance as the SKP is expanded by progressively adding synonymous properties and data instances.

For this purpose, we simulate an ontology engineering process, where an engineer expands the reference ontology by adding synonymous properties and also new data instances retrieved by such properties. The engineer may rank each pair of reference property with a synonymous property by their score, and progress by incrementally add one property at a time. At each iteration it_k, we study (i) the incremental increase ratio (IIR) due to new data instances retrievable by correct synonymous properties added up at it_k and (ii) the incremental noise ratio (INR) due to new data instances retrievable by incorrect synonymous properties added up at it_k. Let $V^+_{sr_i^j} \subset V_{sr_i^j}$ be the values added by correct synonymous properties and $V^-_{sr_i^j} \subset V_{sr_i^j}$ be the values added by incorrect synonymous properties, the calculation of IIR and INR at each iteration k are calculated as

$$IIR_k = \frac{|\bigcup_{i=1}^{k} V^+_{sr_i^j}|}{|\bigcup_{i=1}^{k} (V_{sr_i^j} \cup V_{r_i^j})|} \quad INR_k = \frac{|\bigcup_{i=1}^{k} V^-_{sr_i^j}|}{|\bigcup_{i=1}^{k} (V_{sr_i^j} \cup V_{r_i^j}|}$$

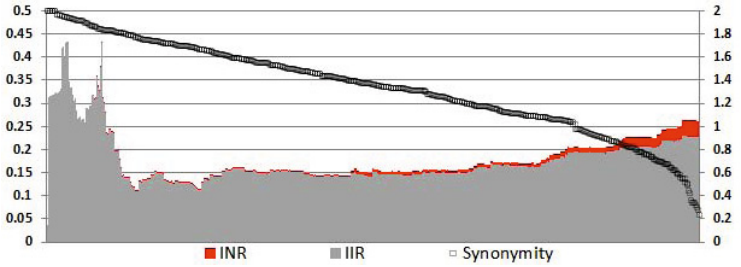


Fig. 6. Incremental Increase Ratio (IIR) and Incremental Noise Ratio (INR) at decreasing synonymity score

Figure 6 shows generally consistent patterns with Figure 5. A high synonymity score rarely introduces errors and when it does, the noise added to the ontology is trivial. As the score drops, noise becomes notable and possibly harms ontology construction.

Error Analysis. To understand the limitations of our method we manually analysed incorrect predictions given by the synonymity measure. We categorise three main sources of errors: (1) highly semantically related properties; (2) property range ambiguity and (3) arguable human annotations.

For many *highly semantically related properties*, we found that often there is a high degree of overlap in their object values. As a result, the synonymity measure makes incorrect predictions based on the data. For example, cities may have the same average temperatures across several months. As a result, our method may predict properties such as "averageTemperatureJune" and "averageTemperatureJuly" to be synonymous. For countries, the property that describes the largest city is considered synonymous with property that describes the capital of the country.

Another source of errors is *property range ambiguity*. We noticed that for some incorrect synonymous property pairs, a common characteristic is that the ranges of the properties derived from data (i.e., the types of the objects of the properties) are ambiguous. Using the synonymous pair *dbo:country* and *dbo:location* for the concept *dbo:MountainRange* as an example, we retrieve the most specific type (ignoring data types) of the objects for each property respectively. We noticed that, *dbo:country* have two distinct ranges in data and the most frequently used is *dbo:Country*, covering 96% of data; while *dbo:location* has 5 distinct ranges and the most frequently used is *dbo:Place*, covering 46% of data, which suggests that the objects of this property is highly inconsistent in terms of their types. Intuitively, if a property's range is ambiguous it would be difficult to assess its synonymity with other properties. We will incorporate this information in our measure in future work.

We also noticed some examples of *highly arguable human annotations*. As an example, *dbo:successor* and *dbpp:after* is predicted synonymous for the class *dbo:Royalty*. However, our annotators considered this example to be incorrect. We manually checked the data and discovered that among all object values (only those that have object types) of *dbpp:after*, 92% belong to the type *dbo:Royalty* and describes a successor of a royalty; and 98% (including *dbo:Royalty*) belong to a class representing a position or person, in which case it describes a successor of certain kind. Thus arguably, although the two properties appear insufficiently synonymous, the data provides additional strong evidence for us to consider them as synonymous for the specific class *dbo:Royalty*.

6 Conclusion and Outlook

In this paper we have introduced the notion of Statistical Knowledge Patterns (SKPs), for capturing the properties used with a certain concept in a bottom-up data-oriented way. We have presented an unsupervised method for generating SKPs, and evaluated it on the DBpedia dataset using the DBpedia ontology as a reference vocabulary. Our evaluation shows that SKPs are able to significantly reduce the number of properties we need to consider, while still maintaining a high coverage of the dataset, compared to considering the complete set of properties present in data. We believe that SKPs are an efficient way to avoid noise in the data, since this is most often present in the "long tail" of property usage. Additionally, the evaluation shows that the clustering of properties, into sets of synonymous properties, allows us to perform query expansion, using the SKP, with high accuracy. Since the methods are completely automated, it is easy to maintain an up-to-date set of SKPs for your dataset, or even across datasets, in order to be able to efficiently query data with sufficiently high recall at any point in time. The main benefits of SKPs include: (1) allowing for both accurate query expansion (through the synonymy of properties) and restriction (through the range axioms); (2) their context dependent nature, describing the usage and meaning of properties in the context of a particular concept and even within a specific dataset; and (3) allowing SKPs to be generated offline (but continuously updated), so that they can be used efficiently at run time.

On an abstract level SKPs can be compared to context-sensitive linguistic resources, such as linguistic frames. Previously, top-down approaches have been used to reengineer linguistic frames, e.g. FrameNet, to KPs. An interesting line of future research is

to integrate them with bottom-up approaches such as EKP and SKP generation. As future work we also plan to focus on the second major feature of the SKPs, namely the new range axioms that were introduced based on observations of data. Just as for the properties themselves, the ranges are also concept-specific. We intend to experimentally validate the range extraction method, and to evaluate the potential of using the range axioms for restricting property selection in query formulation. Other improvements of the method could include taking into account more features of the reference ontology when performing the SKP extraction, e.g. the class hierarchy and additional axioms. We will also generate SKPs for the complete DBpedia ontology[7], and other major sources of Linked Data. In particular, we intend to explore the construction of "cross-dataset" SKPs that include synonymous properties from multiple linked datasets. We then intend to use them in an Information Extraction scenario, for extracting seed data corresponding to the natural language questions of a human user. In such a scenario, SKPs will be an essential component, since they represent the actual properties that are used in data, and since they help to break down the query formulation problem into manageable pieces.

Acknowledgements. Part of this research has been sponsored by the EPSRC funded project LODIE: Linked Open Data for Information Extraction, EP/J019488/1.

References

1. Augenstein, I., Gentile, A.L., Norton, B., Zhang, Z., Ciravegna, F.: Mapping Keywords to Linked Data Resources for Automatic Query Expansion. In: Proc. of the 2nd International Workshop on Knowledge Discovery and Data Mining Meets Linked Open Data (2013)
2. Basse, A., Gandon, F., Mirbel, I., Lo, M.: DFS-based frequent graph pattern extraction to characterize the content of RDF Triple Stores. In: Proceedings of the WebSci 2010: Extending the Frontiers of Society On-Line, Raleigh, NC, US, April 26-27 (2010)
3. Blomqvist, E.: Ontocase-automatic ontology enrichment based on ontology design patterns. In: Bernstein, A., Karger, D.R., Heath, T., Feigenbaum, L., Maynard, D., Motta, E., Thirunarayan, K. (eds.) ISWC 2009. LNCS, vol. 5823, pp. 65–80. Springer, Heidelberg (2009)
4. Budanitsky, A., Hirst, G.: Evaluating WordNet-based Measures of Lexical Semantic Relatedness. Comput. Linguist. 32(1), 13–47 (2006)
5. Cabrio, E., Aprosio, A.P., Cojan, J., Magnini, B., Gandon, F., Lavelli, A.: QAKiS @ QALD-2. In: Proceedings of the ESWC 2012 Workshop Interacting with Linked Data, Heraklion, Greece (2012)
6. Duan, S., Fokoue, A., Hassanzadeh, O., Kementsietsidis, A., Srinivas, K., Ward, M.J.: Instance-Based Matching of Large Ontologies Using Locality-Sensitive Hashing. In: Cudré-Mauroux, P., et al. (eds.) ISWC 2012, Part I. LNCS, vol. 7649, pp. 49–64. Springer, Heidelberg (2012)
7. Gangemi, A., Presutti, V.: Ontology design patterns. In: Staab, S., Studer, R. (eds.) Handbook of Ontologies. International Handbooks on Information Systems, vol. 2, Springer (2009)
8. Gangemi, A., Presutti, V.: Towards a pattern science for the Semantic Web. Semantic Web 1(1-2), 61–68 (2010)

[7] The catalogue will be published at `http://ontologydesignpatterns.org/skp/`

9. Le, N.T., Ichise, R., Le, H.B.: Detecting hidden relations in geographic data. In: Proceedings of the 4th International Conference on Advances in Semantic Processing, pp. 61–68 (2010)
10. Musetti, A., Nuzzolese, A., Draicchio, F., Presutti, V., Blomqvist, E., Gangemi, A., Ciancarini, P.: Aemoo: Exploratory Search based on Knowledge Patterns over the Semantic Web. In: Finalist of the Semantic Web Challenge 2011 (2011)
11. Nuzzolese, A.G., Gangemi, A., Presutti, V., Ciancarini, P.: Encyclopedic knowledge patterns from wikipedia links. In: Aroyo, L., Welty, C., Alani, H., Taylor, J., Bernstein, A., Kagal, L., Noy, N., Blomqvist, E. (eds.) ISWC 2011, Part I. LNCS, vol. 7031, pp. 520–536. Springer, Heidelberg (2011)
12. Parundekar, R., Knoblock, C.A., Ambite, J.L.: Discovering concept coverings in ontologies of linked data sources. In: Cudré-Mauroux, P., et al. (eds.) ISWC 2012, Part I. LNCS, vol. 7649, pp. 427–443. Springer, Heidelberg (2012)
13. Presutti, V., Aroyo, L., Adamou, A., Schopman, B.A.C., Gangemi, A., Schreiber, G.: Extracting Core Knowledge from Linked Data. In: Proc. of the 2nd Intl. Workshop on Consuming Linked Data (COLD 2011), Bonn, Germany, vol. 782, CEUR-WS.org (2011)
14. Presutti, V., Blomqvist, E., Daga, E., Gangemi, A.: Pattern-based ontology design. In: Suárez-Figueroa, M.C., Gómez-Pérez, A., Motta, E., Gangemi, A. (eds.) Ontology Engineering in a Networked World, pp. 35–64. Springer, Heidelberg (2012)
15. Schopman, B., Wang, S., Isaac, A., Schlobach, S.: Instance-Based Ontology Matching by Instance Enrichment. Journal on Data Semantics 1(4), 219–236 (2012)

Complete Query Answering over Horn Ontologies Using a Triple Store

Yujiao Zhou, Yavor Nenov, Bernardo Cuenca Grau, and Ian Horrocks

Department of Computer Science, University of Oxford, UK

Abstract. In our previous work, we showed how a scalable OWL 2 RL reasoner can be used to compute both lower and upper bound query answers over very large datasets and arbitrary OWL 2 ontologies. However, when these bounds do not coincide, there still remain a number of possible answer tuples whose status is not determined. In this paper, we show how in the case of Horn ontologies one can exploit the lower and upper bounds computed by the RL reasoner to efficiently identify a subset of the data and ontology that is large enough to resolve the status of these tuples, yet small enough so that the status can be computed using a fully-fledged OWL 2 reasoner. The resulting hybrid approach has enabled us to compute exact answers to queries over datasets and ontologies where previously only approximate query answering was possible.

1 Introduction

An increasing number of applications rely on RDF and SPARQL for storing and querying semistructured data. The functionality of many such applications is enhanced by an OWL 2 ontology, which is used to *(i)* unambiguously specify the meaning of data in the application, *(ii)* provide the vocabulary and background knowledge needed for users to formulate accurate queries, and *(iii)* enrich query answers with information not explicitly represented in the dataset.

However, the appealing benefits of using an OWL 2 ontology come at the cost of scalability, since answering queries over OWL 2 ontologies is of very high computational complexity. Despite intensive efforts at optimisation, fully-fledged OWL 2 reasoners, such as HermiT [16], Pellet [23] and Racer [9], still fall far short of meeting the scalability demands of applications that require efficient management of large-scale RDF datasets.

To achieve more favourable scalability, a common approach is to delegate reasoning and query answering tasks to a rule-based RDF triple store. State-of-the-art triple stores such as OWLim [3], Oracle's RDF Semantic Graph [26] and RDFox[1] provide robust and scalable query answering support for ontologies in the OWL 2 RL profile [17] and datasets containing millions, or even billions, of triples. However, such triple stores are intrinsically limited in their reasoning capabilities, as they ignore (parts of) axioms in the application's ontology that

[1] `http://www.cs.ox.ac.uk/isg/tools/RDFox/`

H. Alani et al. (Eds.): ISWC 2013, Part I, LNCS 8218, pp. 720–736, 2013.

aren't captured by OWL 2 RL. As a result, they are incomplete: for some combinations of ontology, query and data, they will fail to return all query answers.

In this paper, we propose a novel approach to query answering that addresses the scalability challenge for ontology languages beyond OWL 2 RL without giving up completeness of query answers. The key idea is to employ a hybrid technique that combines an OWL 2 RL reasoner based on a highly scalable RDF triple store (RL reasoner for short) with a fully-fledged OWL 2 reasoner (OWL reasoner for short) such that most of the computational workload can be delegated to the RL reasoner, and the OWL reasoner is used only as necessary to ensure completeness. The difficulty in realising this approach is to efficiently determine when and where fully-fledged reasoning is needed.

Our hybrid query answering technique builds on previous work [27], where we showed how an RL reasoner can be exploited to efficiently compute lower and upper bound query answers over very large datasets and arbitrary OWL 2 ontologies. When the two bounds coincide (which was often the case in the experiments reported in [27]), the query has been fully answered. When the two bounds do not coincide, however, there may still remain a significant number of possible answer tuples whose status is undetermined. In theory, the status of these tuples can be determined using an OWL reasoner, but for large-scale datasets, even checking single tuples is often infeasible in practice.

The main contribution of this paper is a technique for identifying a (typically small) subset of the dataset and ontology that is sufficient for determining the status of a possible answer tuple. The basic idea is that, starting from a query Q and a possible answer tuple $\vec{a}$, we use backward chaining with axioms from the upper-bound ontology to identify those axioms and data triples from the input ontology and dataset that might contribute to a proof that $\vec{a}$ is an answer to Q. An OWL reasoner is then used to check if the identified axioms and data triples entail that $\vec{a}$ is an answer to Q. Currently, our technique is only known to be applicable to Horn ontologies (i.e., ontologies that can be translated to first-order Horn clauses). However, many OWL 2 ontologies are Horn [6], as are all the profiles. Moreover, we conjecture that the approach can be extended to arbitrary OWL 2 ontologies; verifying this conjecture is left for future work.

Our new technique also addresses an important limitation with the approach presented in [27]: if the upper-bound ontology and dataset is unsatisfiable, then it is necessary to check the satisfiability of the input ontology and dataset, but this was impractical with large datasets. Now, we can simply use our new technique to compute the answer to the query `owl:Nothing`(x), with the ontology and dataset being satisfiable iff the answer is empty.

We have developed a reasoner that integrates RDFox and the HermiT OWL reasoner. A preliminary evaluation has shown very promising results. For instance, we can compute in reasonable time the exact answers to a range of queries over the LUBM(40) ontology and dataset—results that are far beyond the capabilities of any other OWL reasoner known to us. Our technique appears to be very effective in identifying small relevant subsets of the data and ontology: for many queries, only 2% of the data and just a few axioms from the ontology

were necessary to determine the status of all unverified answer tuples. Moreover, the effectiveness of our technique is not restricted to LUBM: we have obtained encouraging results with the Fly Anatomy ontology—a biomedical ontology containing more than 7,000 classes and 140,000 axioms—and its associated dataset.

2 Preliminaries

We adopt standard notions from first-order logic with equality, such as variables, constants, terms, atoms, formulas, sentences, substitutions, entailment (written $\models$), and (un)satisfiability. The equality atom between terms t and t' is denoted as $t \approx t'$; we use the abbreviation $t \not\approx t'$ for $\neg(t \approx t')$ (an inequality atom). The falsum atom is denoted as $\bot$ (equivalent to `owl:Nothing`), whereas the dual universal truth atom is denoted as $\top$ (equivalent to `owl:Thing`).

OWL 2 Ontologies. We assume familiarity with the normative specifications of OWL 2 and OWL 2 RL. We deviate slightly from the normative documents only in that we make an explicit distinction between schema-level and data-level axioms. We use *ontology* and *dataset* to refer to a set of schema-level and a set of data-level axioms, respectively. W.l.o.g. we assume that data assertions are given as *facts* (ground atoms), each of which corresponds to a single RDF triple. Consider as our running example the following ontology $\mathcal{O}^{ex}$ and dataset $\mathcal{D}^{ex}$.

$$\begin{aligned}
\mathcal{O}^{ex} = \{&\mathit{SubClassOf}(\mathsf{Animal}\ \mathit{SomeValuesFrom}(\mathsf{eats}\ \mathsf{Thing})), && \text{(T1)}\\
&\mathit{SubClassOf}(\mathsf{Herbivore}\ \mathit{AllValuesFrom}(\mathsf{eats}\ \mathsf{Plant})), && \text{(T2)}\\
&\mathit{DisjointClasses}(\mathsf{Herbivore}\ \mathsf{Carnivore}), && \text{(T3)}\\
&\mathit{SubClassOf}(\mathsf{Carnivore}\ \mathit{MinCardinality}(2\ \mathsf{hasParent}\ \mathsf{Thing}))\} && \text{(T4)}\\
\mathcal{D}^{ex} = \{&\mathit{ClassAssertion}(\mathsf{Animal}\ \mathit{lion}), && \text{(A1)}\\
&\mathit{ClassAssertion}(\mathsf{Animal}\ \mathit{rabbit}), && \text{(A2)}\\
&\mathit{ClassAssertion}(\mathsf{Herbivore}\ \mathit{rabbit}), && \text{(A3)}\\
&\mathit{ClassAssertion}(\mathsf{Herbivore}\ \mathit{sheep}), && \text{(A4)}\\
&\mathit{PropertyAssertion}(\mathsf{eats}\ \mathit{sheep}\ \mathit{grass}), && \text{(A5)}\\
&\mathit{ClassAssertion}(\mathsf{Carnivore}\ \mathit{wolf})\} && \text{(A6)}
\end{aligned}$$

Queries. A *conjunctive query* (CQ) is a first-order formula in the form of $Q(\vec{x}) = \exists\vec{y}(\varphi(\vec{x},\vec{y}))$ with Q a distinguished query predicate and $\varphi(\vec{x},\vec{y})$ a conjunction of atoms without inequalities. The variables in $\vec{x}$ are *distinguished*. The following CQ with a distinguished variables x asks for all individuals that eat plants:

$$Q^{ex}(x) := \exists y(\mathsf{eats}(x,y) \wedge \mathsf{Plant}(y)).$$

A tuple of constants $\vec{a}$ is a *certain answer* to $Q(\vec{x})$ w.r.t. a set of first-order sentences $\mathcal{F}$ and a set of facts $\mathcal{D}$ if $\mathcal{F} \cup \mathcal{D} \models Q(\vec{a})$. The set of all certain answers to $Q(\vec{x})$ w.r.t. $\mathcal{F}$ and $\mathcal{D}$ is denoted as $\mathsf{cert}(Q,\mathcal{F},\mathcal{D})$. For example, the individuals *sheep* and *rabbit* are certain answers to Q^{ex} w.r.t. $\mathcal{O}^{ex}$ and $\mathcal{D}^{ex}$. We omit the

distinguished variables of $Q(\vec{x})$ and write just Q for brevity. *SPARQL conjunctive queries* are CQs with only distinguished variables.[2]

Rule Languages. Rule languages are widely-used knowledge representation formalisms that have strong connections with different fragments of OWL 2 [4]. Specifically, OWL 2 RL is strongly connected to *datalog*, whereas Horn ontologies are related to datalog$^{\pm}$—an extension of datalog with existential quantifiers allowed in rule heads. For instance, our example ontology $\mathcal{O}^{ex}$ is equivalent to the following datalog$^{\pm}$ rules in which all free variables are universally quantified.

$$\exists y(\mathsf{eats}(x, y)) \leftarrow \mathsf{Animal}(x) \tag{P1}$$

$$\mathsf{Plant}(y) \leftarrow \mathsf{eats}(x, y) \wedge \mathsf{Herbivore}(x) \tag{P2}$$

$$\bot \leftarrow \mathsf{Carnivore}(x) \wedge \mathsf{Herbivore}(x) \tag{P3}$$

$$\exists y_1 \exists y_2(\mathsf{hasParent}(x, y_1) \wedge \mathsf{hasParent}(x, y_2) \wedge y_1 \not\approx y_2) \leftarrow \mathsf{Carnivore}(x) \tag{P4}$$

Formally, a datalog$^{\pm}$ rule is a first-order sentence of the following form [5]:

$$\forall \vec{x}(\exists \vec{y}(C_1 \wedge \cdots \wedge C_m) \leftarrow B_1 \wedge \cdots \wedge B_n), \tag{1}$$

where each B_j is an atom with variables in $\vec{x}$ that is neither $\bot$ nor an inequality atom, and either *(i)* $m = 1$ and $C_1 = \bot$, or *(ii)* $m \geq 1$ and, for each $1 \leq i \leq m$, C_i is an atom different from $\bot$ with free variables in $\vec{x} \cup \vec{y}$. A *datalog* rule is a rule of the form (1) with no existentially quantified variables.[3] A *datalog (resp. datalog$^{\pm}$) program* is a set of datalog (resp. datalog$^{\pm}$) rules.

Horn ontologies (i.e., ontologies that can be normalised as a set of first-order Horn clauses) can also be represented by datalog$^{\pm}$ programs. Furthermore, each OWL 2 RL ontology can be represented by a datalog program. Axioms T2 and T3 in our running example are in OWL 2 RL and can be represented by the datalog rules P2 and P3, respectively; in contrast, T1 is outside OWL 2 RL and, as such, is only expressible by a datalog$^{\pm}$ rule (in our case P1).

Datalog rules allow for easy and efficient computation of the dataset $\mathcal{D}_\Sigma$ consisting of all facts entailed by a datalog program Σ and a dataset $\mathcal{D}$. The set $\mathcal{D}_\Sigma$ is called the *materialisation* of Σ w.r.t. $\mathcal{D}$. The set of certain answers $\mathsf{cert}(Q, \Sigma, \mathcal{D})$ for an arbitrary query Q coincides with $\mathsf{cert}(Q, \emptyset, \mathcal{D}_\Sigma)$. Consider, for example, the set Σ_L comprising the datalog rules P2 and P3. The materialisation of Σ_L w.r.t. $\mathcal{D}^{ex}$ extends $\mathcal{D}^{ex}$ with the single fact $\mathit{ClassAssertion}(\mathsf{Plant}\ \mathit{grass})$; clearly, *sheep* is an answer to the query Q^{ex} w.r.t. Σ_L and $\mathcal{D}^{ex}$, but *rabbit* is not.

3 Our Approach in a Nutshell

In this paper, we propose a hybrid reasoning technique that combines an RL reasoner based on a highly scalable RDF triple store with a fully-fledged OWL

[2] In general, a SPARQL query can include non-distinguished variables; however, the semantics of SPARQL means that this is equivalent to treating all variables as distinguished and then applying a suitable projection.

[3] Our definition of datalog is slightly non-standard as it allows conjunction in rule heads; such rules can be equivalently split into multiple rules with atomic heads.

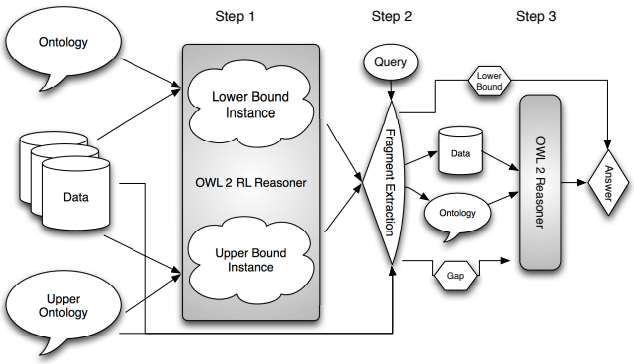


Fig. 1. Overview

reasoner. The key feature of our query answering technique is that it tries to delegate most of the computational workload to the RL reasoner, thus minimising the use of the less scalable OWL reasoner. Given a Horn OWL 2 ontology $\mathcal{O}$, a dataset $\mathcal{D}$, and a CQ Q, we compute the certain answers $\mathsf{cert}(Q, \mathcal{O}, \mathcal{D})$ in three steps, which we summarise next and schematically depict in Figure 1.

Step 1: Lower and Upper Bound Query Answers. Our first step is to compute two OWL 2 RL ontologies $\mathcal{O}_L$ (the *lower bound ontology*) and $\mathcal{O}_U$ (the *upper bound ontology*) satisfying the following property: $\mathsf{cert}(Q, \mathcal{O}_L, \mathcal{D}) \subseteq \mathsf{cert}(Q, \mathcal{O}, \mathcal{D}) \subseteq \mathsf{cert}(Q, \mathcal{O}_U, \mathcal{D})$. Since both $\mathcal{O}_L$ and $\mathcal{O}_U$ are OWL 2 RL ontologies, we can then use an RL reasoner to compute the *lower bound* $\mathsf{cert}(Q, \mathcal{O}_L, \mathcal{D})$ and the *upper bound* $\mathsf{cert}(Q, \mathcal{O}_U, \mathcal{D})$. If the set $G = \mathsf{cert}(Q, \mathcal{O}_U, \mathcal{D}) \backslash \mathsf{cert}(Q, \mathcal{O}_L, \mathcal{D})$ of tuples in the "gap" between lower and upper bound is empty, then the set of certain answers $\mathsf{cert}(Q, \mathcal{O}, \mathcal{D})$ coincides with both lower and upper bounds, in which case we don't need to resort to the OWL reasoner. This step exploits the techniques in our previous work [27], which we briefly recapitulate in Section 4.

Step 2: Computing Ontology and Dataset Fragments. In the second step we exploit the lower and upper bound ontologies and query answers to identify (small) fragments $\mathcal{O}_f$ of $\mathcal{O}$ and $\mathcal{D}_f$ of $\mathcal{D}$ satisfying: $\mathcal{O} \cup \mathcal{D} \models Q(\vec{a})$ iff $\mathcal{O}_f \cup \mathcal{D}_f \models Q(\vec{a})$ for each $\vec{a} \in G$. Thus, $\mathcal{O}_f$ and $\mathcal{D}_f$ are sufficient for determining whether each tuple in G is indeed a certain answer to Q. The fragments $\mathcal{O}_f$ and $\mathcal{D}_f$ depend on both the input query Q and the tuples in G. This novel technique is the main contribution of our paper, and it is described in Section 5.

Step 3: Calling the OWL Reasoner. In the final step we resort to the OWL reasoner to verify whether $\mathcal{O}_f \cup \mathcal{D}_f \models Q(\vec{a})$ for each tuple $\vec{a} \in G$. We return as certain answers the union of the lower bound and the verified tuples in G: $\mathsf{cert}(Q, \mathcal{O}, \mathcal{D}) = \mathsf{cert}(Q, \mathcal{O}_L, \mathcal{D}) \cup \{\vec{a} \in G \mid \mathcal{O}_f \cup \mathcal{D}_f \models Q(\vec{a})\}$.

4 Lower and Upper Bound Query Answers

In our previous work [27] we showed how an RL reasoner can be used to efficiently compute upper and lower bound query answers over arbitrary OWL 2 ontologies.

In this section, we recapitulate the techniques proposed there. Our description will be rather informal, and we refer the interested reader to [27] for details.

Lower Bound Answers. RL reasoners are flexible enough to process arbitrary ontologies on a "best efforts" basis; that is, the reasoner ignores (parts of) the axioms that are outside OWL 2 RL, thus effectively reasoning with a lower bound ontology $\mathcal{O}_L$. RL reasoners are guaranteed to be sound (i.e., $\mathcal{O} \models \mathcal{O}_L$), and hence all the tuples they compute are indeed certain answers; we can therefore compute lower bound answers simply by running the RL reasoner as a "black box" on the input Q, $\mathcal{O}$, and $\mathcal{D}$. For instance, when given our example ontology $\mathcal{O}^{ex}$, dataset $\mathcal{D}^{ex}$, and query Q^{ex}, a typical RL reasoner will reduce $\mathcal{O}^{ex}$ to the OWL 2 RL ontology $\mathcal{O}_L^{ex} = \{\text{T2}, \text{T3}\}$, and will compute $\mathsf{cert}(Q^{ex}, \mathcal{O}_L^{ex}, \mathcal{D}^{ex}) = \{sheep\}$.

Upper Bound Answers. We transform $\mathcal{O}$ into an OWL 2 RL ontology $\mathcal{O}_U$ such that $\mathcal{O}_U \models \mathcal{O}$. First, $\mathcal{O}$ is normalised into a datalog$^{\pm}$ program $\Sigma^{\pm}$ using a variant of the structural transformation of first-order logic (see [16,27]). For instance, our example ontology $\mathcal{O}^{ex}$ can be normalised into the datalog$^{\pm}$ program consisting of rules P1-P4. The crucial second step is the transformation of the resulting datalog$^{\pm}$ program into a (stronger) datalog program Σ_U satisfying $\Sigma_U \models \mathcal{O}$; roughly speaking, Σ_U is obtained by Skolemising all existential quantifiers into fresh constants. For example, the datalog$^{\pm}$ rules P1 and P4 get transformed into the rules D1, D4 and D5 to give the following datalog program:

$$
\begin{aligned}
\Sigma_U^{ex} = \{ & \mathsf{eats}(x, c) \leftarrow \mathsf{Animal}(x), && \text{(D1)}\\
& \mathsf{Plant}(y) \leftarrow \mathsf{eats}(x, y) \wedge \mathsf{Herbivore}(x), && \text{(D2)}\\
& \bot \leftarrow \mathsf{Carnivore}(x) \wedge \mathsf{Herbivore}(x), && \text{(D3)}\\
& \mathsf{hasParent}(x, c_1) \wedge \mathsf{hasParent}(x, c_2) \leftarrow \mathsf{Carnivore}(x), && \text{(D4)}\\
& \bot \leftarrow c_1 \approx c_2\}. && \text{(D5)}
\end{aligned}
$$

Finally, the datalog program Σ_U is transformed into the upper bound OWL 2 RL ontology $\mathcal{O}_U$, where $\mathcal{O}_U \models \Sigma_U$; roughly speaking, each rule in Σ_U is transformed into an OWL 2 RL axiom by "rolling up" the rule's body and head into class descriptions, while possibly introducing fresh predicates in order to satisfy the syntactic restrictions of OWL 2 RL. For instance, the datalog rules D1–D5 in our running example are transformed into the following OWL 2 RL axioms:

$$
\begin{aligned}
\mathcal{O}_U^{ex} = \{ & \mathit{SubClassOf}(\mathsf{Animal}\ \mathit{HasValue}(\mathsf{eats}\ c)), && \text{(R1)}\\
& \mathit{SubClassOf}(\mathsf{Herbivore}\ \mathit{AllValuesFrom}(\mathsf{eats}\ \mathsf{Plant})), && \text{(R2)}\\
& \mathit{DisjointClasses}(\mathsf{Herbivore}\ \mathsf{Carnivore}), && \text{(R3)}\\
& \mathit{SubClassOf}(\mathsf{Carnivore}\ \mathit{HasValue}(\mathsf{hasParent}\ c_1)), && \text{(R4.1)}\\
& \mathit{SubClassOf}(\mathsf{Carnivore}\ \mathit{HasValue}(\mathsf{hasParent}\ c_2)), && \text{(R4.2)}\\
& \mathit{DifferentFrom}(c_1\ c_2)\}. && \text{(R4.3)}
\end{aligned}
$$

As a result, we obtain that $\mathcal{O}_U \models \mathcal{O}$ and hence $\mathsf{cert}(Q, \mathcal{O}, \mathcal{D}) \subseteq \mathsf{cert}(Q, \mathcal{O}_U, \mathcal{D})$. Clearly, the transformation of $\mathcal{O}$ into the upper bound ontology $\mathcal{O}_U$ will in general introduce consequences that are not entailed by the original ontology $\mathcal{O}$.

To see this, consider again our running example. The axioms R1, R2, A1 and A2 entail $\mathit{ObjectPropertyAssertion}(\mathsf{eats}\ \mathit{rabbit}\ c)$, $\mathit{ObjectPropertyAssertion}(\mathsf{eats}\ \mathit{lion}\ c)$ and $\mathit{ClassAssertion}(\mathsf{Plant}\ c)$. We thus get that (in addition to *sheep*) *rabbit* and *lion* are also answers to Q^{ex}, i.e. $\mathsf{cert}(Q^{ex}, \mathcal{O}_U^{ex}, \mathcal{D}^{ex}) = \{\mathit{sheep}, \mathit{rabbit}, \mathit{lion}\}$. However, we have that $\mathit{lion} \notin \mathsf{cert}(Q^{ex}, \mathcal{O}^{ex}, \mathcal{D}^{ex})$.

The final transformation from Σ_U into $\mathcal{O}_U$ is only required if the RL reasoner to be used only accepts OWL 2 RL ontologies; our RL reasoner RDFox can handle datalog rules natively, and this transformation can be dispensed with.

Dealing with Unsatisfiability. An important limitation with the approach presented in [27] is that, given an ontology $\mathcal{O}$ and a dataset $\mathcal{D}$, if $\mathcal{O}_L \cup \mathcal{D}$ is satisfiable (i.e., $\mathsf{cert}(\bot(x), \mathcal{O}_L, \mathcal{D}) = \emptyset$) but $\mathcal{O}_U \cup \mathcal{D}$ is unsatisfiable (i.e., $\mathsf{cert}(\bot(x), \mathcal{O}_U, \mathcal{D}) \neq \emptyset$), we must check if $\mathcal{O} \cup \mathcal{D}$ is satisfiable; if $\mathcal{O} \cup \mathcal{D}$ is satisfiable, we can still use the above procedure to compute upper bound answers, but if $\mathcal{O} \cup \mathcal{D}$ is *not* satisfiable, then everything is entailed and the (upper bound) answer to any query is trivially the set of all tuples of the appropriate arity that can be formed from individuals in $\mathcal{D}$. The difficulty is that, when $\mathcal{D}$ is large, it may be impractical to check the satisfiability of $\mathcal{O} \cup \mathcal{D}$ using an OWL reasoner.

We can now address this issue by using our new hybrid query answering technique: if $\mathsf{cert}(\bot(x), \mathcal{O}_L, \mathcal{D}) = \emptyset$, but $\mathsf{cert}(\bot(x), \mathcal{O}_U, \mathcal{D}) \neq \emptyset$, then in Step 2 we will compute fragments $\mathcal{O}_f$ and $\mathcal{D}_f$ for $\bot(x)$, and in Step 3 we will use these fragments with an OWL reasoner to compute $\mathsf{cert}(\bot(x), \mathcal{O}, \mathcal{D})$. Clearly, $\mathcal{O} \cup \mathcal{D}$ is satisfiable iff $\mathsf{cert}(\bot(x), \mathcal{O}, \mathcal{D}) = \emptyset$.

5 Computing Ontology and Dataset Fragments

Given an input ontology $\mathcal{O}$, a dataset $\mathcal{D}$ and a set of possible answer tuples G, our goal is to compute small ontology $\mathcal{O}_f \subseteq \mathcal{O}$ and dataset $\mathcal{D}_f \subseteq \mathcal{D}$ such that $\mathcal{O}_f \cup \mathcal{D}_f \models Q(\vec{a})$ iff $\mathcal{O} \cup \mathcal{D} \models Q(\vec{a})$ for each tuple $\vec{a} \in G$.

5.1 Overview

In a nutshell, our technique for computing $\mathcal{O}_f$ and $\mathcal{D}_f$ works as follows.

1. We consider the upper bound datalog rules Σ_U and, for each $\vec{a} \in G$, we compute all (minimal) proofs of $Q(\vec{a})$ in $\Sigma_U \cup \mathcal{D}$. Specifically, we consider "backward chaining" proofs based on SLD-resolution.
2. We define $\mathcal{D}_f$ (resp. Σ_f) as the set of facts in $\mathcal{D}$ (resp. rules in Σ_U) that have been used in some SLD-resolution proof for some $\vec{a} \in G$.
3. Finally, we "trace back" the rules in $\Sigma_f \subseteq \Sigma_U$ to the OWL 2 axioms $\mathcal{O}_f \subseteq \mathcal{O}$ from which they were derived.

To illustrate this process, let us consider our example ontology $\mathcal{O}^{ex}$, data set $\mathcal{D}^{ex}$ and query Q^{ex}. In this case, we have $\{\mathit{sheep}\}$ as the lower bound answer and $\{\mathit{sheep}, \mathit{rabbit}, \mathit{lion}\}$ as the upper bound answer; our goal is thus to determine whether *rabbit* and *lion* are indeed certain answers. To this end, we consider the upper bound datalog program Σ_U^{ex}, and inspect all the "backward chaining"

proofs for $Q^{ex}(rabbit)$ and $Q^{ex}(lion)$ in $\Sigma_U^{ex} \cup \mathcal{D}^{ex}$. Consider for example the following proof of $Q^{ex}(rabbit)$.

$$
\begin{aligned}
&S_0 := \mathsf{eats}(rabbit, y) \wedge \mathsf{Plant}(y) && \\
&S_1 := \mathsf{Plant}(c) \wedge \mathsf{Animal}(rabbit) && \text{via D1} \\
&S_2 := \mathsf{Animal}(rabbit) \wedge \mathsf{eats}(x, c) \wedge \mathsf{Herbivore}(x) && \text{via D2} \\
&S_3 := \mathsf{eats}(x, c) \wedge \mathsf{Herbivore}(x) && \text{via A2} \\
&S_4 := \mathsf{Herbivore}(x) \wedge \mathsf{Animal}(x) && \text{via D1} \\
&S_5 := \mathsf{Animal}(rabbit) && \text{via A3} \\
&S_6 := \top && \text{via A2}
\end{aligned}
$$

Starting from the goal $Q^{ex}(rabbit) = \mathsf{eats}(rabbit, y) \wedge \mathsf{Plant}(\mathsf{y})$, we can use D1 and the unifier $\{x \mapsto rabbit, y \mapsto c\}$ to obtain the subgoal S_1, which, together with D1, entails S_0. Then, we can use rule D2 and the unifier $\{y \mapsto c\}$ to obtain from S_1 the new subgoal S_2. The first conjunct in S_2 can be eliminated using the fact A2 in $\mathcal{D}^{ex}$ to produce S_3. From S_3 we can use again rule D1 to produce S_4. The first conjunct in S_4 can be eliminated using fact A3 in $\mathcal{D}^{ex}$ and finally we can obtain the empty goal by subsequently using fact A2 to eliminate the remaining atom. We have now shown that $\{\text{D1}, \text{D2}, \text{A2}, \text{A3}\} \models Q^{ex}(rabbit)$; therefore, facts A2 and A3 must be included in $\mathcal{D}_f$, and axioms T1 and T2 from $\mathcal{O}^{ex}$, from which rules D1 and D2 were (respectively) derived, must be included in $\mathcal{O}_f$.

To identify all the axioms and facts in $\mathcal{O} \cup \mathcal{D}$ that are relevant to $Q^{ex}(rabbit)$ and $Q^{ex}(lion)$, we need to consider all their possible backward chaining proofs. By doing so, we can show that only axioms T1 and T2, and facts A1, A2 and A3 are (possibly) relevant to determining the status of *rabbit* and *lion*.

5.2 Technical Approach

We start by formalising backward chaining proofs based on SLD-resolution.

Definition 1. *A* goal *is a conjunction of function-free atoms $A_1 \wedge \ldots \wedge A_m$. The* SLD-resolution rule *takes as premises a goal and a datalog rule, and it produces a new goal as follows*

$$\frac{A_1 \wedge \ldots \wedge A_m, \qquad C_1 \wedge \ldots \wedge C_q \leftarrow B_1 \wedge \ldots \wedge B_p}{A_2\theta \wedge \ldots \wedge A_m\theta \wedge B_1\theta \wedge \ldots, B_p\theta}$$

where θ is the most general unifier of A_1 and C_j for some $1 \leq j \leq q$. The new goal, together with the rule entail the original goal. An SLD-proof *of a goal G_0 in a datalog program Γ is a sequence of goals $G_0 \overset{r_1,\theta_1}{\rightsquigarrow} G_1 \rightsquigarrow \ldots \rightsquigarrow G_{n-1} \overset{r_n,\theta_n}{\rightsquigarrow} G_n$, where $G_n = \top$ and each G_{i+1} is obtained from G_i and rule $r_{i+1} \in \Gamma$ by means of a single SLD-resolution with substitution θ_{i+1}. Finally, we say that a rule r is* relevant *for G_0 in Γ if there exists an SLD-proof of G_0 in Γ involving r.*

SLD-resolution is sound and complete for datalog: for each datalog program Γ, CQ $Q(\vec{x}) = \exists \vec{y}(\varphi(\vec{x}, \vec{y}))$ and tuple of constants $\vec{a}$ we have $\Gamma \models Q(\vec{a})$ iff there exists an SLD-proof of the goal $\varphi(\vec{a}, \vec{y})$ in Γ.

We are now ready to define the relevant fragments $\mathcal{O}_f \subseteq \mathcal{O}$ and $\mathcal{D}_f \subseteq \mathcal{D}$ that we can use to verify the answers in G using an OWL reasoner.

Definition 2. *Let Q, $\mathcal{O}$ and $\mathcal{D}$ be the input CQ, Horn ontology and dataset respectively, let Σ_U be the upper bound datalog program for $\mathcal{O}$, and let $\Xi(\cdot)$ be the function mapping each axiom in $\mathcal{O}$ into its corresponding set of rules in Σ_U. Finally, let G be the set of tuples between the lower and upper bounds. The (Q, G)-*relevant fragments *$\mathcal{O}_f$ of $\mathcal{O}$ and $\mathcal{D}_f$ of $\mathcal{D}$ are defined as follows $\mathcal{O}_f = \{\alpha \in \mathcal{O} \mid \exists \vec{a} \in G$ and $\exists r \in \Xi(\alpha)$ s.t. r is relevant for $Q(\vec{a})$ in $\Sigma_U \cup \mathcal{D}\}$, $\mathcal{D}_f = \{\alpha \in \mathcal{D} \mid \exists \vec{a} \in G$ s.t. α is relevant for $Q(\vec{a})$ in $\Sigma_U \cup \mathcal{D}\}$.*

The correctness of our approach is established by the following theorem.

Theorem 1. *Let $\mathcal{O}_f$ and $\mathcal{D}_f$ be the (Q, G)-relevant fragments of $\mathcal{O}$ and $\mathcal{D}$, respectively. Then, $\mathcal{O} \cup \mathcal{D} \models Q(\vec{a})$ iff $\mathcal{O}_f \cup \mathcal{D}_f \models Q(\vec{a})$ for each $\vec{a} \in G$.*

The full proof of Theorem 1 can be found in our accompanying technical report.[4] The idea behind it is, however, quite straightforward. The 'if' direction follows directly from the fact that $\mathcal{O}_f \cup \mathcal{D}_f \subseteq \mathcal{O} \cup \mathcal{D}$. For the 'only if' direction, assume that $\mathcal{O} \cup \mathcal{D} \models Q(\vec{a})$. From $\Sigma_U \models \mathcal{O}$ it follows that $\Sigma_U \cup \mathcal{D} \models Q(\vec{a})$. By the completeness of SLD-resolution, there exists an SLD proof of $Q(\vec{a})$ in $\Sigma_U \cup \mathcal{D}$, and let R be the set of rules used in this proof. By construction, the axioms in $\mathcal{O}$ and facts in $\mathcal{D}$ that correspond to rules in R are contained in $\mathcal{O}_f$ and $\mathcal{D}_f$ respectively, and it can be further shown that these axioms entail $Q(\vec{a})$.

5.3 An Optimised Backward Chaining Algorithm

Computing all SLD-proofs for each answer in the gap between lower and upper bounds can be infeasible in practice with a naive backward chaining algorithm. Indeed, our problem is more challenging than typical backward chaining reasoning in datalog, where computing just a single proof suffices to verify the goal.

In this section, we describe an optimised algorithm for computing the set $R_{\vec{a}}$ of all rules that appear in an SLD proof of $Q(\vec{a})$ in $\Sigma_U \cup \mathcal{D}$, where the facts in $\mathcal{D}$ are treated as rules with an empty body, i.e., of the form $A(\vec{a}) \leftarrow$. To ensure termination of backward chaining, we apply the well-known tabling technique [24,22]. To improve performance, we use an aggressive pruning technique that exploits the upper and lower bound ontologies to detect irrelevant branches in the backward chaining tree. In the remainder of this section, we describe the specifics of our implementation of backward chaining.

Backward Chaining with Tabling. Our implementation of backward chaining with tabling is based on the techniques described in [22]. We deviate from [22] in that our algorithm only terminates once all SLD-proofs of the goal are computed, and also in that we keep track of all rules used in such proofs.

We first describe our data structures. To keep track of all SLD proofs of a goal $Q(\vec{a})$ and all rules that occur in them, we maintain a labelled tree t_A for each

[4] `http://tinyurl.com/bl5unv6`

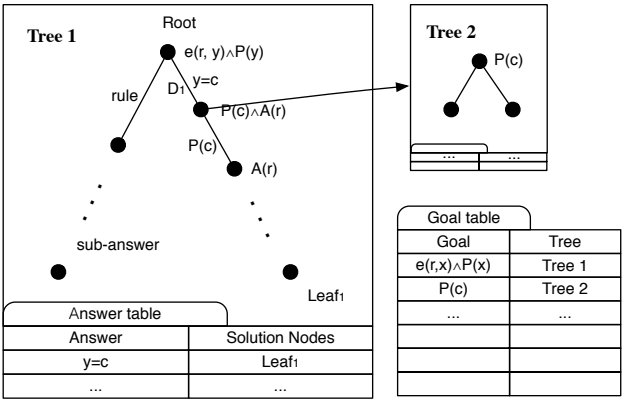


Fig. 2. Data Structure

encountered subgoal A consisting of a single atom; additionally, we maintain a similar tree for the original goal $Q(\vec{a})$ (see Figure 2). The tree t_A encodes all proofs of A; each node of t_A is labelled with a goal and each edge of t_A is labelled with a pair of a substitution and a datalog rule. The labels of each edge and the nodes that it connects encode an SLD resolution step, and a branch in t_A encodes an SLD derivation. We associate with t_A an *answer table*, which will eventually map each grounding of the root goal that proves it to the list of all relevant leaf nodes in t_A for that grounding. We also maintain a global *goal table* mapping each relevant subgoal to its corresponding tree (c.f. Figure 2). Finally, we say that a node u with associated goal $A_1 \wedge \ldots \wedge A_n$ is *linked* to a tree t if the first atom A_1 in the goal corresponds to the root of t (e.g. node D_1 in Figure 2 is linked to Tree 2); we use the goal table to check whether u is linked to t.

The backward chaining algorithm is initialised with a tree $t_{Q(\vec{a})}$ consisting of just a root node labelled with the goal $Q(\vec{a})$; then, we add an entry in the Goal table mapping $Q(\vec{a})$ to $t_{Q(\vec{a})}$ and we associate with $t_{Q(\vec{a})}$ an empty answer table. After this initialisation step, we evaluate the root node of $t_{Q(\vec{a})}$ using the recursive procedure described below, which takes as input a node u in a tree t.

Case 1. If u is labelled with $\top$, we have reached a proof of the goal $A(\vec{x})$ labelling the root of t. We take $A(\vec{a})$ to be the grounding of $A(\vec{x})$ using the composition of the substitutions on the path between the root of t and the node u. We add u to the list associated with $A(\vec{a})$ in the answer table of t. We then try to resolve $A(\vec{a})$ with the goal of every node that is linked to t. If we can resolve $A(\vec{a})$ with the goal of such a node v using a substitution θ, we add a child node v' to v, we label v with the resolvent, and we label the edge between v and v' with the pair $\langle A(\vec{a}), \theta \rangle$. We then recursively evaluate the node v'.

Case 2. If u is the root node of t, we resolve its goal with all possible rules, we create a child node for each of the resolvents, label the new nodes and edges accordingly, and recursively evaluate each of the child nodes.

Case 3. Otherwise, let $A_1 \wedge \ldots \wedge A_m$ be the goal of u.

3.1. If A_1 has not been tabled yet, i.e. it is not in the goal table, we initialise a tree t' with root node v, label v with A_1, add an entry $\langle A_1, t' \rangle$ to the goal table linking u to t, and recursively process v;

3.2. otherwise, we retrieve the available answers for A_1 from its associated tree, resolve u with those answers, create a child node for each of the resolvents, and recursively compute each added node.

Pruning. To improve performance, we apply a pruning technique that exploits the lower and upper bound ontologies. Let v be a node with associated goal $Q_v := A_1 \wedge \ldots \wedge A_m$. Before recursively evaluating v, we proceed as follows.

- If $\mathsf{cert}(Q_v, \mathcal{O}_U, \mathcal{D}) = \emptyset$, we terminate the evaluation of the current node as this branch cannot lead to a proof. This is due to the fact that the rules used in the backward chaining algorithm are logically equivalent to $\mathcal{O}_U \cup \mathcal{D}$.
- If Q_v contains no variables, and $\mathsf{cert}(Q_v, \mathcal{O}_L, \mathcal{D}) \neq \emptyset$, we create a child node for v, label it with $\top$, and we label the new edge with the empty substitution and the set of atoms $A_1, \ldots, A_m$. We recursively evaluate the new node, after which we terminate the evaluation of v. We can do so because $\mathcal{O}_L \cup \mathcal{D} \models Q_v$, and we know that the current branch will lead to exactly one proof.
- Otherwise, if A_1 contains no variable and $\mathsf{cert}(A_1, \mathcal{O}_L, \mathcal{D}) \neq \emptyset$, we create a child node for v, we label it with the goal $A_2 \wedge \ldots \wedge A_m$, label the new edge with the empty substitution and the atom A_1, and recursively evaluate the new node, after which we terminate the evaluation of v. We can do so because $\mathcal{O}_L \cup \mathcal{D} \models A_1$, and because v has no other children.

Rule Extraction. The backward-chaining evaluation of the goal $Q(\vec{a})$ results in a forest of trees encoding all possible proofs of $Q(\vec{a})$ in $\Sigma_U \cup \mathcal{D}$. In addition to all proofs of $Q(\vec{a})$, the forest also contains many superfluous derivations that should be ignored. We now describe an algorithm that traverses the forest and extracts the set $R_{\vec{a}}$ of all rules that participate in proofs of $Q(\vec{a})$. The algorithm builds the set $R_{\vec{a}}$ by carrying out a bottom-up, breath-first search on the nodes in the forest whose goals appear in proofs of $Q(\vec{a})$. It proceeds as follows.

Step 1 Initialise a set N with all solution nodes in the answer table of $t_{Q(\vec{a})}$.

Step 2 While N is not empty, remove from N a node v with a goal $A_1 \wedge \cdots \wedge A_m$. If v has a parent, do the following:

2.1. Add the parent of v to N.

2.2. If v is a resolvent of its parent and a rule $r \in \Sigma_U \cup \mathcal{D}$, add r to $R_{\vec{a}}$.

2.3. If v is a resolvent of its parent and an answer A from a tree t, retrieve all solution nodes for A in t and add them to N.

6 Evaluation

We have developed a prototype reasoner to carry out a preliminary evaluation. Our prototype integrates the RL reasoner RDFox[5] and an OWL reasoner, which

[5] `http://www.cs.ox.ac.uk/isg/tools/RDFox/`

Table 1. Statistics for datasets

Data	DL	Horn	Existential	Classes	Properties	Axioms	Individuals	Dataset
LUBM(n)	$\mathcal{SHI}$	Yes	8	43	32	93	$1.7 \times 10^4 n$	$10^5 n$
FLY	$\mathcal{SRI}$	Yes	8,396	7,533	24	144,407	1,606	6,308

Table 2. Results for LUBM(40)

Query	$\lvert V\rvert$	n	$\lvert G\rvert$	t_f	$\lvert\mathcal{O}_f\rvert$	$\lvert\mathcal{D}_f\rvert$	t_{check}	t_{total}
M_1	2	3	39	36.4	6	29041	H: 23.3	H: 60.7
M_2	3	4	1	37.1	6	29004	H: 4.0	H: 42.1
M_3	4	6	16	38.2	6	29054	H: 8.4	H: 47.6
M_4	2	3	30	36.0	6	29032	H: 23.3	H: 60.2
M_5	3	4	4	39.4	6	29010	H: 24.0	H: 64.3
M_6	4	6	29	2,845.8	10	87209	H: 483.0	H: 3339.4
M_7	3	5	15	38.0	6	29033	H: 10.3	H: 49.3
M_8	3	5	14	39.3	6	29038	H: 11.9	H: 52.2
M_9	3	4	10	328.9	12	86785	H: 556.2	H: 886.7
S	1	2	39	310.0	12	86802	H: 1,780.0 P: 16,592.1	H: 2,126.5 P: 16,870.0

in our case can be either HermiT[16] or Pellet[23]. RDFox is used to compute lower and upper bound query answers (c.f. Step 1 in Section 3), as well as to assist with pruning during backward chaining (c.f. Section 5.3). The OWL reasoner is used with the fragments computed by the backward chaining algorithm (c.f. Step 3 in Section 3) to determine the status of any tuples in the gap.

RDFox is a in-memory triple store that supports OWL 2 RL and datalog reasoning, and uses shared memory parallel reasoning for increased efficiency and scalability. An important feature of RDFox is its rapid query response time—this is particularly relevant during backward chaining, where queries are used in a crucial pruning optimisation. HermiT and Pellet are well-established OWL reasoners that provide support for CQ answering. HermiT can answer tree-shaped CQs with a single answer variable; Pellet supports SPARQL CQs.

In our experiments we have used LUBM and the Fly Anatomy ontology as test sets; their key features are summarised in Table 1. All tests were performed on a 14 core 3.30GHz Intel Xeon E5-2643 with 125GB of RAM, and running Linux 2.6.32. All times are given in seconds.

Evaluation Results for LUBM. LUBM is a well-known benchmark ontology that comes with a predefined dataset generator parameterised by the number of universities. We tested our reasoner on LUBM(40), which contains in its dataset over 4 million facts about 40 universities. We used the 14 standard LUBM queries as well as 78 synthetic queries generated using SyGENiA [12].

Using RDFox, we were able to compute lower and upper bound answers for all 92 queries in less than 20s (c.f. Step 1 in Section 3). As the focus of this paper is on checking the tuples in the gap between the two bounds of a given query,

Table 3. Results for FLY

Query	$\|V\|$	n	$\|G\|$	t_f	$\|\mathcal{O}_f\|$	$\|\mathcal{D}_f\|$	t_{check} (H)	t_{total}	t_{HermiT}
Q_1	2	3	803	108.9	224	4515	45.9	155.2	3,465.9
Q_2	3	5	342	97.7	224	4054	16.0	114.0	3,179.0
Q_3	1	1	28	91.0	217	3712	0.9	92.3	5,863.3
Q_4	2	3	25	94.3	233	3762	4.7	99.2	2,944.3
Q_5	2	2	518	100.3	222	3712	24.0	124.6	3,243.7

we concentrate our attention on those queries whose bounds do not coincide. Only 6 queries show a non-empty gap ($Q_3, Q_{45}, Q_{51}, Q_{64}, Q_{67}, Q_{69}$)—all of them SyGENiA generated—and, due to the relatively simple nature of the LUBM ontology, it is inevitable that these queries look very similar. Since the generated queries tend to produce unrealistically large answers, we added some additional terms to those queries in order to make them more specific and thus return smaller answers. The resulting 10 "non-trivial" queries, denoted by M_1–M_9 and S, are all tree-shaped CQs, with S being the only SPARQL CQ.[6] Performance on these queries is summarised in Table 2, where $|V|$, n and $|G|$ denote the number of variables, the number of triple patterns and the number of gap tuples for each query respectively; t_f, $|\mathcal{O}_f|$ and $|\mathcal{D}_f|$ denote the time needed to compute the relevant fragments $\mathcal{O}_f$ and $\mathcal{D}_f$ using backward chaining, and their respective sizes; t_{check} denotes the total time required for checking all the tuples in G using the OWL reasoner (c.f. Step 3 in Section 3); and t_{total} denotes the total time for answering the query. We have presented timings for Hermit (H) on all queries, and Pellet (P) on the single SPARQL CQ.

We can observe that backward chaining times were rather modest for all queries but M_6, for which the computed backward chaining tree had a very large branching factor. We can also observe that for all queries the dataset fragment $\mathcal{D}_f$ contains only about 2% of the facts in LUBM(40), while $\mathcal{O}_f$ contained just a few schema-level axioms. This significant reduction in size made it possible for HermiT to verify answer tuples in reasonable time; indeed, query answering for LUBM(40) is far beyond the capabilities of HermiT or Pellet, and we were unable to verify even a single answer tuple using either OWL reasoner over the original ontology and dataset. A standard optimisation applied by RL reasoners such as OWLim is to classify the original ontology first and add the entailed subsumption axioms in $\mathcal{O}_L$. Although this optimisation closes the gap between the lower and upper bounds for query S, allowing, for example, OWLim to compute all answers for S, it has no effect on queries M_1–M_9.

To test the scalability of our reasoner, we have also evaluated queries M_1, M_6, M_9 and S against the datasets LUBM(1)–LUBM(40) (results for queries M_2–M_5, M_7 and M_8 are similar to those for M_1). The results of the evaluation are summarised in Figure 3, which shows the timings and memory usage of our reasoner for the different datasets.

[6] All test queries are available at `http://tinyurl.com/ccmwvc6`.

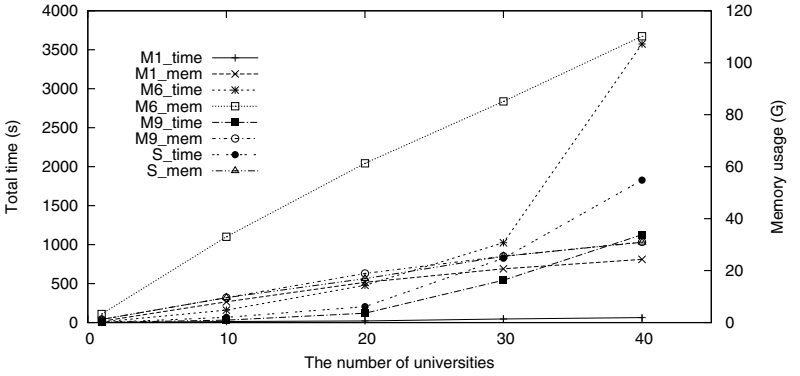


Fig. 3. Scalability test for LUBM(1)-LUBM(40)

Evaluation Results for FLY. Fly Anatomy is a realistic and complex ontology describing the anatomy of flies, which comes with a dataset containing more than 6,000 facts. We have tested our system using 5 realistic queries provided by the domain experts who are developing the ontology; all are CQs with non-distinguished variables, so we were only able to use HermiT in the evaluation.

RDFox was able to compute lower and upper bound answers for all 5 queries in less than 15s, with the bounds being different in all cases. Our results are summarised in Table 3; columns 1–9 are the same as in Table 2, and column 10 gives the time taken for HermiT to answer the query—in contrast to the case of LUBM(40), HermiT is able to answer all these queries directly. In each case, $\mathcal{O}_f$ contained less than 0.2% of all the schema-level axioms; in contrast, $\mathcal{D}_f$ contained about 60% of the facts in the dataset. The reduction in number of schema-level axioms had a significant effect on performance: our reasoner took less than 200s per query, whereas HermiT required more than $3,000s$ per query.

7 Related Work

In recent years there has been a growing interest in the problem of query answering over ontologies and large-scale datasets. Some OWL 2 reasoners, such as HermiT, Pellet and RACER, support query answering, but despite intensive efforts at optimisation they can only deal with modestly-sized datasets [14,15,10].

The idea of using a rule-based engine for query answering over ontologies in the description logic $\mathcal{SHIQ}$ was proposed by Hustadt et al. [11] and implemented in the KAON2 system. The transformation of the ontology, however, results in a disjunctive datalog program which is exponential in the worst case. An alternative approach based on tableaux reasoning and data summarisation was implemented in the reasoner SHER, which is complete for the description logic $\mathcal{SHIN}$, but (effectively) supports only SPARQL CQs [7].

Many specialised query answering techniques have been developed for ontologies in the QL and RL profiles of OWL 2. RL reasoners such as OWLim,

Oracle's and RDFox are based on forward-chaining reasoning. QL reasoners such as QuOnto [1], Presto [21], and Quest [20] are based on query rewriting. These reasoners are, however, incomplete for ontologies outside the relevant profile. Query rewriting techniques have been extended to more expressive Horn Description Logics, and implemented in systems such as REQUIEM [19] and Clipper [8].

The idea of combining a profile-specific reasoner with a fully-fledged OWL 2 reasoner was proposed in [2], but only for ontology classification. The idea of transforming the ontology, data, and/or query to obtain upper bound query answers has also received some attention in the Semantic Web literature. In addition to our own previous work [27], approximations into OWL 2 QL [13,18] and into Datalog [25] have also been explored; however, all these techniques are worst case exponential, and the question of how to deal with cases where upper and lower bounds do not coincide was not considered.

8 Conclusion

In this paper we have described a hybrid approach for complete query answering over Horn OWL 2 ontologies. Our technique combines a scalable OWL 2 RL reasoner with a fully-fledged OWL 2 reasoner s.t. most of the computational workload is delegated to the RL reasoner, with the OWL 2 reasoner being used only as necessary to ensure completeness. We have implemented a prototype reasoner that integrates the RL reasoner RDFox and the OWL 2 reasoner HermiT. A preliminary evaluation of our prototype produced very promising results: we managed to compute in reasonable time the exact answers to a range of queries over LUBM(40)—results that are far beyond the capabilities of any other OWL 2 reasoner known to us. Our system also outperforms HermiT on the realistic Fly ontology by at least an order of magnitude. We are currently working on an extension to support query answering over arbitrary OWL 2 ontologies (and not just Horn ontologies), as well as on several promising optimisations.

Acknowledgements. The research was supported by the EPSRC funded ExODA and Score! projects, the Royal Society, and the EU FP7 Optique project.

References

1. Acciarri, A., Calvanese, D., Giacomo, G.D., Lembo, D., Lenzerini, M., Palmieri, M., Rosati, R.: Quonto: Querying ontologies. In: AAAI. pp. 1670–1671 (2005)
2. Armas Romero, A., Cuenca Grau, B., Horrocks, I.: MORe: Modular combination of owl reasoners for ontology classification. In: Cudré-Mauroux, P., et al. (eds.) ISWC 2012, Part I. LNCS, vol. 7649, pp. 1–16. Springer, Heidelberg (2012)
3. Bishop, B., Kiryakov, A., Ognyanoff, D., Peikov, I., Tashev, Z., Velkov, R.: OWLim: A family of scalable semantic repositories. Semantic Web J. 2(1), 33–42 (2011)
4. Bry, F., Eisinger, N., Eiter, T., Furche, T., Gottlob, G., Ley, C., Linse, B., Pichler, R., Wei, F.: Foundations of rule-based query answering. In: RR. pp. 1–153 (2007)

5. Cali, A., Gottlob, G., Lukasiewicz, T., Marnette, B., Pieris, A.: Datalog+/-: A family of logical knowledge representation and query languages for new applications. In: LICS (2010)
6. Cuenca Grau, B., Horrocks, I., Krötzsch, M., Kupke, C., Magka, D., Motik, B., Wang, Z.: Acyclicity conditions and their application to query answering in description logics. In: Proc. of the 13th Int. Conf. on Principles of Knowledge Representation and Reasoning, KR 2012 (2012), download/2012/CHKKMMW12a.pdf
7. Dolby, J., Fokoue, A., Kalyanpur, A., Ma, L., Schonberg, E., Srinivas, K., Sun, X.: Scalable grounded conjunctive query evaluation over large and expressive knowledge bases. In: Sheth, A.P., Staab, S., Dean, M., Paolucci, M., Maynard, D., Finin, T., Thirunarayan, K. (eds.) ISWC 2008. LNCS, vol. 5318, pp. 403–418. Springer, Heidelberg (2008)
8. Eiter, T., Ortiz, M., Šimkus, M., Tran, T.K., Xiao, G.: Query rewriting for horn-shiq plus rules. In: AAAI (2012)
9. Haarslev, V., Möller, R.: RACER system description. J. of Automated Reasoning (JAR), 701–705 (2001)
10. Haarslev, V., Hidde, K., Möller, R., Wessel, M.: The RacerPro knowledge representation and reasoning system. Semantic Web 3(3), 267–277 (2012)
11. Hustadt, U., Motik, B., Sattler, U.: Reasoning in Description Logics by a Reduction to Disjunctive Datalog. Journal of Automated Reasoning 39(3), 351–384 (2007)
12. Imprialou, M., Stoilos, G., Grau, B.: Benchmarking ontology-based query rewriting systems. In: Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence. AAAI (2012)
13. Kaplunova, A., Möller, R., Wandelt, S., Wessel, M.: Towards scalable instance retrieval over ontologies. In: Bi, Y., Williams, M.-A. (eds.) KSEM 2010. LNCS, vol. 6291, pp. 436–448. Springer, Heidelberg (2010)
14. Kollia, I., Glimm, B.: Cost based query ordering over OWL ontologies. In: Cudré-Mauroux, P., et al. (eds.) ISWC 2012, Part I. LNCS, vol. 7649, pp. 231–246. Springer, Heidelberg (2012)
15. Kollia, I., Glimm, B., Horrocks, I.: SPARQL query answering over OWL ontologies. In: Antoniou, G., Grobelnik, M., Simperl, E., Parsia, B., Plexousakis, D., De Leenheer, P., Pan, J. (eds.) ESWC 2011, Part I. LNCS, vol. 6643, pp. 382–396. Springer, Heidelberg (2011)
16. Motik, B., Shearer, R., Horrocks, I.: Hypertableau reasoning for description logics. J. of Artificial Intelligence Research (JAIR) 36(1), 165–228 (2009)
17. Motik, B., Cuenca Grau, B., Horrocks, I., Wu, Z., Fokoue, A., Lutz, C.: OWL 2 Web Ontology Language Profiles. In: W3C Recommendation, 2nd edn. (2012)
18. Pan, J., Thomas, E., Zhao, Y.: Completeness guaranteed approximations for OWL-DL query answering. DL 477 (2009)
19. Pérez-Urbina, H., Horrocks, I., Motik, B.: Efficient query answering for OWL 2. In: Bernstein, A., Karger, D.R., Heath, T., Feigenbaum, L., Maynard, D., Motta, E., Thirunarayan, K. (eds.) ISWC 2009. LNCS, vol. 5823, pp. 489–504. Springer, Heidelberg (2009)
20. Rodriguez-Muro, M., Calvanese, D.: High performance query answering over DL-Lite ontologies. In: KR (2012)
21. Rosati, R.: Prexto: Query rewriting under extensional constraints in DL-Lite. In: Simperl, E., Cimiano, P., Polleres, A., Corcho, O., Presutti, V. (eds.) ESWC 2012. LNCS, vol. 7295, pp. 360–374. Springer, Heidelberg (2012)
22. Sagonas, K., Swift, T.: An abstract machine for tabled execution of fixed-order stratified logic programs. ACM Transactions on Programming Languages and Systems (TOPLAS) 20(3), 586–634 (1998)

23. Sirin, E., Parsia, B., Cuenca Grau, B., Kalyanpur, A., Katz, Y.: Pellet: A practical OWL-DL reasoner. J. Web Semantics (JWS) 5(2), 51–53 (2007)
24. Tamaki, H., Sato, T.: OLD resolution with tabulation. In: Shapiro, E. (ed.) ICLP 1986. LNCS, vol. 225, pp. 84–98. Springer, Heidelberg (1986)
25. Tserendorj, T., Rudolph, S., Krötzsch, M., Hitzler, P.: Approximate OWL-reasoning with screech. In: Calvanese, D., Lausen, G. (eds.) RR 2008. LNCS, vol. 5341, pp. 165–180. Springer, Heidelberg (2008)
26. Wu, Z., Eadon, G., Das, S., Chong, E.I., Kolovski, V., Annamalai, M., Srinivasan, J.: Implementing an inference engine for RDFS/OWL constructs and user-defined rules in Oracle. In: ICDE, pp. 1239–1248 (2008)
27. Zhou, Y., Cuenca Grau, B., Horrocks, I., Wu, Z., Banerjee, J.: Making the Most of your Triple Store: Query Answering in OWL 2 Using an RL Reasoner. In: WWW (2013)

Author Index