Philippe Cudré-Mauroux Jeff Heflin
Evren Sirin Tania Tudorache
Jérôme Euzenat Manfred Hauswirth
Josiane Xavier Parreira Jim Hendler
Guus Schreiber Abraham Bernstein
Eva Blomqvist (Eds.)

LNCS 7649

The Semantic Web – ISWC 2012

11th International Semantic Web Conference
Boston, MA, USA, November 2012
Proceedings, Part I

1 Part I

Springer

Philippe Cudré-Mauroux Jeff Heflin
Evren Sirin Tania Tudorache
Jérôme Euzenat Manfred Hauswirth
Josiane Xavier Parreira Jim Hendler
Guus Schreiber Abraham Bernstein
Eva Blomqvist (Eds.)

The Semantic Web – ISWC 2012

11th International Semantic Web Conference
Boston, MA, USA, November 11-15, 2012
Proceedings, Part I

Springer

Volume Editors

Philippe Cudré-Mauroux; University of Fribourg, Switzerland; pcm@unifr.ch

Jeff Heflin; Lehigh University, USA; heflin@cse.lehigh.edu

Evren Sirin; Clark & Parsia, USA; evren@clarkparsia.com

Tania Tudorache; Stanford University, USA; tudorache@stanford.edu

Jérôme Euzenat; INRIA & LIG, France; jerome.euzenat@inria.fr

Manfred Hauswirth; DERI, NUI Galway, Ireland; manfred.hauswirth@deri.org

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

Jim Hendler; Rensselaer Polytechnic Institute (RPI), USA; hendler@cs.rpi.edu

Guus Schreiber; VU University Amsterdam, The Netherlands; guus.schreiber@vu.nl

Abraham Bernstein; University of Zurich, Switzerland; bernstein@ifi.uzh.ch

Eva Blomqvist; Linköping University, Sweden; eva.blomqvist@liu.se

ISSN 0302-9743 e-ISSN 1611-3349
ISBN 978-3-642-35175-4 e-ISBN 978-3-642-35176-1
DOI 10.1007/978-3-642-35176-1
Springer Heidelberg Dordrecht London New York

Library of Congress Control Number: 2012952320

CR Subject Classification (1998): I.2.3-4, H.2.3-5, H.2.1, H.2.8, I.2.6, H.3.4-5, D.2.12, D.3.2, I.7.2, H.5.3-4, 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

The Semantic Web has come a long way. What started as a vision of a machine-readable Web over ten years ago now consists of a vibrant community of researchers, practitioners, enthusiast, companies, and, finally, users. Topics that were once cutting-edge research have now arrived in the mainstream and have even become part of political agendas. The sharing of public information in the form of linked data has become a major argument for the transparency of administrations, and institutions around the globe are putting their data online. Companies from various sectors such as the BBC, Google, IBM, or *The New York Times* release products that are based on Semantic Web technologies. Against all prophecies of failure, the Semantic Web is flourishing.

The International Semantic Web Conference is the premier forum for Semantic Web research, where cutting-edge scientific results and technological innovations are presented, where problems and solutions are discussed, and where the future of this vision is being developed. It brings together specialists in fields such as artificial intelligence, databases, social networks, distributed computing, Web engineering, information systems, human–computer interaction, natural language processing, and the social sciences for tutorials, workshops, presentations, keynotes, and sufficient time to have detailed discussions.

This volume contains the main proceedings of the 11th International Semantic Web Conference (ISWC 2012), which was held in Boston, USA, in November 2012. Even though the economic times are anything but easy we received tremendous response to our calls for papers from a truly international community of both researchers and practitioners. Every paper was thoroughly evaluated following practices appropriate for its track and its evaluation measure. The breadth and scope of the papers finally selected for inclusion in this volume speak to the quality of the conference and to the contributions made by researchers whose work is presented in these proceedings. As such, we were all honored and proud that we were invited to serve the community in the stewardship of this edition of ISWC.

The Research Track of the conference attracted 186 submissions, 41 of which were accepted, resulting in a 22% acceptance rate. Each paper received at least three, and sometimes as many as five, reviews from members of the Program Committee. 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 10-hour 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.

As the Semantic Web develops, we find a changing variety of subjects that emerge. This year the keywords of accepted papers were distributed as follows (frequency in parentheses): knowledge representation and reasoning (13), querying the Semantic Web and database technologies (10), ontology engineering (7), machine learning and information extraction (7), data mining and analysis (6), ontology mapping (6), linked data (5), languages, tools and methodologies (4), interacting with Semantic Web data (4), instance mapping (4), evaluation (4), social and emergent semantics (4), cleaning, assurance, and provenance (4), search and information retrieval (3), federated/distributed systems (3), scalable systems (3), Semantic Web services (3), exploiting the social Web (3), knowledge acquisition (2), natural language processing (2), query languages (2), uncertainty (2), modeling users and contexts (2), semantic streams and sensors (2), ontology learning (1), user interfaces (1), mashing up data and processes (1), trust, privacy and security (1), and personalized access (1).

This edition of the International Semantic Web Conference marks the introduction 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 35 submissions from all areas of the Semantic Web: including reasoning, querying, searching, matching, and annotating. Papers were of two main categories, namely, evaluation (comparing several approaches to a problem) and corpus analysis. To our surprise, testing a hypothesis through an experiment was not explicitly considered. We also received very few papers aiming at reproducing existing experiments. Eight papers were accepted, corresponding to a 23% acceptance rate. Each paper was reviewed by at least three members of the Program Committee paying special attention to the reproducibility criteria. In spite of the limited number of accepted papers, they address a large range of areas, such as linked stream data, federated query processing, tag recommendation, entity summarization, and OWL reasoning.

The Semantic Web In-Use Track received 77 submissions. At least three members of the In-Use Program Committee provided reviews for each paper. Seventeen papers were accepted – a 22% acceptance rate. The large number of submissions this year demonstrated the increasingly diverse breadth of applications of Semantic Web technologies in practice. The papers demonstrated how semantic technologies are applied in a variety of domains, including eGovernment, smart cities, biomedicine, or question answering. Several papers dealt with applying reasoning for a variety of use cases, while others dealt with streaming data and processing complex events. A number of infrastructure papers contributed to the state of art for Linked Open Data and for querying large data sets. Very exciting application papers demonstrated how semantic technologies are applied in diverse ways, starting from using linked data in mobile environments to employing full-fledged artificial intelligence methods in real-time use cases.

The Doctoral Consortium is a key event at the ISWC conference. PhD students in the Semantic Web field get an opportunity to present their thesis proposals and to interact with leading academic and industrial scientists in the field, who act as their mentors. Out of 21 submissions to the Doctoral Consortium, six were accepted as for presentation at the conference. For discussion at the special Consortium-only session on 12 November, nine additional proposals were selected. The Doctoral Consortium day is organized as a highly interactive event, in which students present their proposals and receive extensive feedback and comments from the mentors as well as from their peers.

A unique aspect of the ISWC conference is the Semantic Web Challenge, now in its 10th year, with the goal of demonstrating practical progress toward achieving the vision of the Semantic Web. Organized this year by Diana Maynard and Andreas Harth, the competition enables practitioners and scientists to showcase leading-edge real-world applications of Semantic Web technology.

The keynote talks given by leading scientists or practitioners in their field further enriched the ISWC program. Thomas W. Malone, the director of the Center for Collective Intelligence at the Massachusetts Institute of Technology, discussed the phenomenon of collective intelligence and how it interrelates with the Semantic Web. Jeanne Holm, an evangelist for data.gov, discussed the changing global landscape of data sharing and the role the Semantic Web is playing in it. Mark Musen, a professor of medicine of the Stanford Center for Biomedical Informatics Research, discussed how the fundamental challenges of AI are still with us and await embracing to fulfill the vision of the Semantic Web. And last but not least, Nigel Shadbolt, Deputy Head of the School of Electronics and Computer Science at the University of Southampton, gave a lively dinner talk.

As in previous ISWC editions, the conference included an extensive Tutorial and Workshop program. Claudia d'Amato and Thomas Scharrenbach, the Chairs of this track, selected a stellar and diverse collection of 9 tutorials and 18 workshops, where the only problem that the participants faced was which of the many exciting workshops to attend. Workshops and tutorials were chosen on the ground of two different but complementary criteria: maintaining the history of the most promising, challenging, and highly attended workshops such as the Ontology Matching Workshop, the Consuming Linked Data Workshop, the Ontology Patterns Workshop, or the Uncertainty Reasoning for the Semantic Web Workshop and highlighting the attention on new, challenging, and visionary research trends as testified by the Programming the Semantic Web Workshop, the Semantic Sensor Network Workshop, the Web of Linked Entities Workshop, the Semantic Technologies Applied to Biomedical Informatics and Individualized Medicine Workshop, the Web of Data for E-Commerce Tutorial, the Machine Learning for Linked Data Tutorial, the Linked Data for Development Tutorial, or the Financial Information Management using the Semantic Web Tutorial. Also, particular attention was dedicated to the heterogeneity and scalability issues and related aspects, which explains the choice of the Workshop on Large and Heterogeneous Data and Quantitative Formalization in the Semantic Web, the Tutorial on RDF Query Processing in the Cloud, and the Tutorial on Scalable Semantic Processing of Hudge, Distributed Real-Time Streams.

We would like to thank Birte Glimm and David Huynh for organizing a lively Poster and Demo Session. As in 2011, the Posters and Demos were introduced in a Minute Madness Session, where every presenter got 60 seconds to provide a teaser for their poster or demo.

Ivan Herman, Tom Heath, and Tim Berners-Lee coordinated a top-flight Industry Track where end-users of Semantic Web and Linked Data technologies shared their "warts and all" experiences with the research community. The track attracted presentations from enterprises of all scales, from startups through to software, hardware, and retail giants such as Oracle, Cray, Cisco, EMC, and BestBuy.

We are indebted to Eva Blomqvist, our Proceedings Chair, who provided invaluable support in compiling the volume that you now hold in your hands (or see on your screen) and exhibited superhuman patience in allowing the other Chairs to stretch deadlines to the absolute limits. Many thanks to Jen Golbeck, the Fellowship Chair, for securing and managing the distribution of student travel grants and thus helping students who might not have otherwise attended the conference to come to Boston. Peter Mika and David Wood were tireless in their work as Sponsorship Chairs, knocking on every conceivable virtual "door" and ensuring an unprecedented level of sponsorship this year. We are especially grateful to all the sponsors for their generosity.

As has been the case in the past, ISWC 2012 also contributed to the linked data cloud by providing semantically annotated data about many aspects of the conference. This contribution would not have been possible without the efforts of Li Ding our Metadata Chair. Oshani Seneviratne, our Publicity Chair, was tirelessly twittering and sending old-fashioned (and highly appreciated) announcements on the mailing lists, creating far more lively "buzz" than ISWC ever had.

Our very special thanks go to the Local Organization Team, led by Lalana Kagal. She did an outstanding job of handling local arrangements, thinking of every potential complication way before it arose, often doing things when members of the Organizing Committee were only beginning to think about asking for them. We managed to juggle so many balls, that some of us were dizzy just looking at it. Special thanks go to the staff of MIT conference services – Cathi Di Iulio Levine, Nicole Silva, Lynne Alyson Lenker, and Eva Cabone – for their enormous resourcefulness, foresight, and anticipation of the conference needs and requirements. Also many thanks for the designers at the University of Zurich Multimedia and e-Learning Services, who provided all the design work often going beyond the call of any duty.

Finally, we would like to thank all members of the ISWC Organizing Committee not only for handling their tracks superbly, but also for their wider contribution to the collaborative decision-making process in organizing the conference.

September 2012

Philippe Cudré-Mauroux
Jeff Heflin
Program Committee Co-chairs
Research Track

Manfred Hauswirth
Josie Xavier Perreira
Jérôme Euzenat
Program Committee Co-chairs
Evaluation Track

Tania Tudorache
Evren Sirin
Program Committee Co-chairs
Semantic Web In-Use Track

Claudia d'Amato
Thomas Scharrenbach
Workshop & Tutorials Co-chairs

Jim Hendler
Guus Schreiber
Doctoral Consortium Chairs

Abraham Bernstein
Conference Chair

Conference Organization

Organizing Committee

General Chair

Abraham Bernstein	University of Zurich, Switzerland

Local Chair

Lalana Kagal	Massachusetts Institute of Technology, USA

Program Chairs–Research Track

Philippe Cudré-Mauroux	University of Fribourg, Switzerland
Jeff Heflin	Lehigh University, USA

In-Use Track Chairs

Tania Tudorache	Stanford University, USA
Evren Sirin	Clark&Parsia, USA

Evaluations and Experiments Track Chairs

Jérôme Euzenat	INRIA & LIG, France
Manfred Hauswirth	DERI, NUI Galway, Ireland
Josiane Xavier Parreira	DERI, NUI Galway, Ireland

Industry Track Chair

Tim Berners-Lee	W3C, USA
Tom Heath	Talis, UK
Ivan Herman	W3C & CWI, The Netherlands

Doctoral Consortium Chairs

Guus Schreiber	VU University Amsterdam, The Netherlands
Jim Hendler	RPI, USA

Fellowship Chair

Jen Golbeck	University of Maryland, USA

Posters and Demos Chairs

David Huynh	Google, USA
Birte Glimm	University of Ulm, Germany

Workshops and Tutorials Chairs

Claudia d'Amato	University of Bari, Italy
Thomas Scharrenbach	University of Zurich, Switzerland

Semantic Web Challenge Chairs

Diana Maynard	University of Sheffield, UK
Andreas Harth	KIT, Germany

Sponsorship Chairs

Peter Mika	Yahoo! Research, Spain
David Wood	3 Round Stones Inc., USA

Metadata Chair

Li Ding	Qualcomm, USA

Webmaster

Stéphane Corlosquet	MGH, USA

Proceedings Chair

Eva Blomqvist	Linköping University, Sweden

Publicity Chair

Oshani Seneviratne	MIT, USA

Senior Program Committee–Research

Harith Alani	The Open University, UK
Lora Aroyo	Vrije Universiteit Amsterdam, The Netherlands
Jérôme Euzenat	INRIA, France
Andreas Harth	Karlsruhe Institute of Technology, Germany
Pascal Hitzler	Wright State University, USA
Ian Horrocks	University of Oxford, UK
Lalana Kagal	MIT, USA
David Karger	MIT, USA
Manolis Koubarakis	National and Kapodistrian University of Athens, Greece
Diana Maynard	University of Sheffield, UK
Deborah L. McGuinness	RPI, USA
Natasha F. Noy	Stanford University, USA
Peter Patel-Schneider	Nuance Communications, USA

Axel Polleres	Siemens AG Österreich, Austria & DERI, Galway, Ireland
Heiner Stuckenschmidt	University of Mannheim, Germany
Jie Tang	Tsinghua University, China
Hai Zhuge	Chinese Academy of Sciences, China

Program Committee–Research

Karl Aberer
José Luis Ambite
Grigoris Antoniou
Kemafor Anyanwu
Sören Auer
Jie Bao
Michael Benedikt
Sonia Bergamaschi
Christian Bizer
Eva Blomqvist
Paolo Bouquet
John Breslin
Dan Brickley
Paul Buitelaar
Diego Calvanese
Huajun Chen
Key-Sun Choi
Benoit Christophe
Vassilis Christophides
Fabio Ciravegna
Lin Clark
Oscar Corcho
Isabel Cruz
Bernardo Cuenca Grau
Carlo Curino
Richard Cyganiak
Claudia d'Amato
Mathieu d'Aquin
David De Roure
Mike Dean
Stefan Decker
Renaud Delbru
Gianluca Demartini
Ian Dickinson
Stefan Dietze
Li Ding
John Domingue
Miriam Fernandez
Tim Finin
Giorgos Flouris
Achille Fokoue
Enrico Franconi
Bo Fu
Avigdor Gal
Fabien Gandon
Aldo Gangemi
Birte Glimm
Jennifer Golbeck
Alasdair J. G. Gray
Paul Groth
Tudor Groza
Michael Gruninger
Nicola Guarino
Christophe Guéret
Gerd Gröner
Volker Haarslev
Harry Halpin
Siegfried Handschuh
Michael Hausenblas
Sandro Hawke
Tom Heath
Martin Hepp
Michiel Hildebrand
Aidan Hogan
Vasant Honavar
Matthew Horridge
Andreas Hotho
Wei Hu
Arantza Illarramendi
Krzysztof Janowicz
Yannis Kalfoglou
Anastasios Kementsietsidis
Brian Kettler
Pavel Klinov

Craig Knoblock
Adila A. Krisnadhi
Markus Krötzsch
Freddy Lecue
Alexander Löser
Christopher Matheus
Yutaka Matsuo
Peter Mika
Prasenjit Mitra
Luc Moreau
Boris Motik
Yohei Murakami
Wolfgang Nejdl
Yuan Ni
Jeff Pan
Bijan Parsia
Alexandre Passant
Paulo Pinheiro Da Silva
Dimitris Plexousakis
Alexandra Poulovassilis
Valentina Presutti
Abir Qasem
Guilin Qi
Yuzhong Qu
Marie-Christine Rousset
Sebastian Rudolph
Marta Sabou
Satya Sahoo
Manuel Salvadores
Bernhard Schandl
Francois Scharffe
Stefan Schlobach
Daniel Schwabe
Juan F. Sequeda
Luciano Serafini
Amit Sheth
Pavel Shvaiko
Wolf Siberski
Elena Simperl
Michael Sintek
Spiros Skiadopoulos
Kavitha Srinivas
Steffen Staab
Umberto Straccia
Rudi Studer
York Sure-Vetter
Valentina Tamma
Letizia Tanca
Kerry Taylor
Sergio Tessaris
Philippe Thiran
Thanh Tran
Raphael Troncy
Christos Tryfonopoulos
Tania Tudorache
Frank Van Harmelen
Maria Esther Vidal
Haofen Wang
Josiane Xavier Parreira
Lina Zhou
Esteban Zimanyi
Antoine Zimmermann

Additional Reviewers–Research

Alan Akbik
Pramod Anantharam
Manuel Atencia
George Baryannis
Domenico Beneventano
Jürgen Bock
Georgeta Bordea
Mohamed Boukhebouze
Stephane Campinas
Michele Catasta
Pierre-Antoine Champin
Francesco Corcoglioniti
Evangelia Daskalaki
Brian Davis
Chiara Di Francescomarino
Jianfeng Du
Songyun Duan
Jinan El-Hachem
Philipp Frischmuth
Sidan Gao
Andrés García-Silva
Paolo Garza

Mouzhi Ge
Francesco Guerra
Dalkandura Gunaratna
Robert Gwadera
Lushan Han
Holmer Hemsen
Daniel M. Herzig
Aidan Hogan
Robert Isele
Nophadol Jekjantuk
Pavan Kapanipathi
Ernesto Jimenez-Ruiz
Pavan Kapanipathi
Yevgeny Kazakov
Hyeongsik Kim
Ilianna Kollia
Haridimos Kondylakis
Kostis Kyzirakos
John Liagouris
Chang Liu
Diana Maynard
Varish Mulwad
Rammohan Narendula
Nadeschda Nikitina
Charalampos Nikolaou
Shohei Ohsawa
Alexandra Olteanu
Fabrizio Orlandi
Mirko Orsini
Matteo Palmonari
Evan Patton
Sean Policarpio
Behrang Qasemizadeh
Emanuele Rabosio
Paraskevi Raftopoulou
Padmashree Ravindra
Timothy Redmond
Yuan Ren
Martin Rezk
Benedicto Rodriguez-Castro
Mariano Rodriguez-Muro
Simon Scheider
Michael Schneider
Jennifer Sleeman
Serena Sorrentino
Jie Tang
Vinhtuan Thai
Joerg Unbehauen
Larysa Visengeriyeva
Jens Wissmann
Zhixian Yan
Jun Zhao
Yuting Zhao
Dmitriy Zheleznyakov

Program Committee–Semantic Web In-Use

Jans Aasman
Anupriya Ankolekar
Sören Auer
Christian Bizer
Cecile Bothorel
Ivan Cantador
Vinay Chaudhri
Oscar Corcho
Gianluca Correndo
Mathieu D'Aquin
Mike Dean
Leigh Dodds
Federico Michele Facca
Hugh Glaser
Mark Greaves
Tudor Groza
Peter Haase
Armin Haller
Siegfried Handschuh
Michael Hausenblas
Tom Heath
Ivan Herman
Pascal Hitzler
Rinke Hoekstra
Matthew Horridge
Wei Hu
Krzysztof Janowicz
Pavel Klinov
Matthias Klusch
Jens Lehmann

Yuan-Fang Li
Thorsten Liebig
Adrian Mocan
Lyndon Nixon
Vit Novacek
Natasha Noy
Massimo Paolucci
Alexandre Passant
Carlos Pedrinaci
Héctor Pérez-Urbina
Yves Raimond
Cartic Ramakrishnan
Marco Rospocher
Matthew Rowe
Marta Sabou
Manuel Salvadores
Patrick Sinclair
Milan Stankovic
Nenad Stojanovic
Markus Strohmaier
Jamie Taylor
Giovanni Tummarello
Michael Uschold
Willem Robert van Hage
Jacco Van Ossenbruggen
Holger Wache
Kewen Wang
Shenghui Wang
Zhe Wu
Fouad Zablith
Amal Zouaq

Additional Reviewers–In-Use

Michelle Cheatham
Brian Davis
Laura Dragan
Cosmin Dumitru
Christian Huetter
Prateek Jain
Christoph Lange
Akshay Maan
Marc Schaaf
Michael Schmidt
William Smith
Claus Stadler
Jesse Wang
Amapali Zaveri

Program Committee–Evaluations and Experiments

Denilson Barbosa
Payam Barnaghi
Oscar Corcho
Thierry Declerck
Renaud Delbru
Jérôme Euzenat
Raúl García-Castro
Asunción Gómez-Pérez
Siegfried Handschuh
Andreas Harth
Alois Haselboeck
Manfred Hauswirth
Conor Hayes
Ryutaro Ichise
Antoine Isaac
Manolis Koubarakis
Thomas Krennwallner
Christian Meilicke
Marta Sabou
Sherif Sakr
Kai-Uwe Sattler
Francois Scharffe
Ondrej Svab-Zamazal
Martin Theobald
Christos Tryfonopoulos
Giovanni Tummarello
Willem Robert Van Hage
Maria Esther Vidal
Josiane Xavier Parreira
Shenghui Wang

Additional Reviewers–Evaluations and Experiments

Nitish Aggarwal
Stephane Campinas
Davide Ceolin
Brian Davis
Laura Dragan
Aidan Hogan
Ioana Hulpus
Vit Novacek
Magdalena Ortiz
Paraskevi Raftopoulou
Patrik Schneider
Guohui Xiao

Program Committee–Doctoral Consortium

Harith Alani
Oscar Gorcho
Fausto Guinchiliglia
Ian Horrocks
Diana Maynard
Natasha Noy
Elena Simperl
Steffen Staab
Frank van Harmelen

Sponsors

Gold Sponsors	Silver Sponsors
AI Journal	IBM
fluid Operations	Pitney Bowes
Oracle	
systap	
Yahoo! Research	

Driving Innovation with Open Data and Interoperability
(Keynote Talk)

Jeanne Holm

Data.gov, U.S. General Services Administration
jeanne.m.holm@jpl.nasa.gov

Abstract

Data.gov, a flagship open government project from the US government, opens and shares data to improve government efficiency and drive innovation. Sharing such data allows us to make rich comparisons that could never be made before and helps us to better understand the data and support decision making. The adoption of open linked data, vocabularies and ontologies, the work of the W3C, and semantic technologies is helping to drive Data.gov and US data forward. This session will help us to better understand the changing global landscape of data sharing and the role the semantic web is playing in it.

This session highlights specific data sharing examples of solving mission problems from NASA, the White House, and many other governments agencies and citizen innovators.

The Semantic Web and Collective Intelligence
(Keynote Talk)

Thomas Malone

MIT Sloan School of Management, Cambridge, MA
malone@mit.edu

Abstract

The original vision of the Semantic Web was to encode semantic content on the web in a form with which machines can reason. But in the last few years, we've seen many new Internet-based applications (such as Wikipedia, Linux, and prediction markets) where the key reasoning is done, not by machines, but by large groups of people.

This talk will show how a relatively small set of design patterns can help understand a wide variety of these examples. Each design pattern is useful in different conditions, and the patterns can be combined in different ways to create different kinds of collective intelligence. Building on this foundation, the talk will consider how the Semantic Web might contribute to–and benefit from–these more human-intensive forms of collective intelligence.

Tackling Climate Change: Unfinished Business from the Last "Winter"

(Keynote Talk)

Mark A. Musen

Stanford Center for Biomedical Informatics Research, Stanford University
musen@stanford.edu

Abstract

In the 1990s, as the World Wide Web became not only world wide but also dense and ubiquitous, workers in the artificial intelligence community were drawn to the possibility that the Web could provide the foundation for a new kind of AI. Having survived the AI Winter of the 1980s, the opportunities that they saw in the largest, most interconnected computing platform imaginable were obviously compelling. With the subsequent success of the Semantic Web, however, our community seems to have stopped talking about many of the issues that researchers believe led to the AI Winter in the first place: the cognitive challenges in debugging and maintaining complex systems, the drift in the meanings ascribed to symbols, the situated nature of knowledge, the fundamental difficulty of creating robust models. These challenges are still with us; we cannot wish them away with appeals to the open-world assumption or to the law of large numbers. Embracing these challenges will allow us to expand the scope of our science and our practice, and will help to bring us closer to the ultimate vision of the Semantic Web.

Table of Contents – Part I

Table of Contents – Part II

Doctoral Consortium – Short Papers

MORe: Modular Combination of OWL Reasoners for Ontology Classification

Ana Armas Romero, Bernardo Cuenca Grau, and Ian Horrocks

Department of Computer Science, University of Oxford

Abstract. Classification is a fundamental reasoning task in ontology design, and there is currently a wide range of reasoners highly optimised for classification of OWL 2 ontologies. There are also several reasoners that are complete for restricted fragments of OWL 2 , such as the OWL 2 EL profile. These reasoners are much more efficient than fully-fledged OWL 2 reasoners, but they are not complete for ontologies containing (even if just a few) axioms outside the relevant fragment. In this paper, we propose a novel classification technique that combines an OWL 2 reasoner and an efficient reasoner for a given fragment in such a way that the bulk of the workload is assigned to the latter. Reasoners are combined in a black-box modular manner, and the specifics of their implementation (and even of their reasoning technique) are irrelevant to our approach.

1 Introduction

Classification—the problem of identifying the subsumption relationships between all pairs of classes in the input ontology—is a fundamental reasoning task in ontology design. For expressive ontology languages, however, the decision problems associated with classification have a very high worst-case complexity; in particular, subsumption with respect to an OWL 2 ontology is known to be a 2NEXPTIME-complete problem [15,6].

In spite of these discouraging complexity results, highly optimised reasoners such as Pellet [21], FaCT++ [22], RacerPro [10] and HermiT [8] are able to classify many ontologies used in applications. The optimisations employed by these reasoners aim not only to improve performance on individual subsumption tests, but also to reduce the number of tests performed when classifying a given ontology—most OWL 2 reasoners use variants of the well-known Enhanced Traversal Algorithm to incrementally construct a compact representation of the subsumption relation, along with the *told subsumptions* optimisation, which provides an inexpensive way of identifying "obvious" subsumption relationships that hold in the input ontology [2,11]. Identifying obvious non-subsumptions is also important, as most possible subsumption relationships do not hold, and has been addressed by optimisations such as *completely defined concepts*, which identifies a fragment of the ontology for which told subsumptions provide complete information, *model-merging*, and other related techniques that exploit the computations performed during individual class satisfiability tests [23,11,9,8].

However, notwithstanding extensive and ongoing research into optimisation techniques, the classification of large ontologies—such as the SNOMED medical ontology—can still require a very large number of subsumption tests, and even if no individual test is very costly, the total amount of time required for classification can still be large. This (and other performance issues) has motivated a growing interest in so-called *lightweight description logics*: weaker logics that enjoy favourable computational properties. OWL 2 includes several *profiles* (language fragments) based on such lightweight DLs, including OWL 2 EL, a profile based on the $\mathcal{EL}^{++}$ DL for which most standard reasoning tasks can be performed in polynomial time [19]. Very efficient profile-specific reasoners have been developed for OWL 2 EL, including CEL [3] and ELK [16], which can classify ontologies as large as SNOMED in just a few seconds.

Unfortunately, a reasoner for profile $\mathcal{L}$ (an $\mathcal{L}$-reasoner) is only able to (completely) classify ontologies in the $\mathcal{L}$ profile ($\mathcal{L}$-ontologies), and restricting the ontology to a given profile may be undesirable or even infeasible in practice, with many existing ontologies falling outside all of the tractable profiles of OWL 2. In many cases, however, such ontologies contain only a relatively small number of axioms that are outside one of the tractable fragments. For example, of the 219,224 axioms in the latest version of the National Cancer Institute Ontology (NCI), only 65 are outside the OWL 2 EL profile. Using a suitable $\mathcal{L}$-reasoner to efficiently classify most classes (i.e., to find all their subsumers) in the signature (or vocabulary) of a given ontology might, therefore, lead to significant improvements in performance. Unfortunately, using an $\mathcal{L}$-reasoner in this way is far from simple, as even a single axiom that is outside $\mathcal{L}$ could affect every class in the ontology.

In this paper, we propose a novel technique where an efficient $\mathcal{L}$-reasoner and a fully fledged OWL 2 reasoner are combined in a modular way to classify an OWL 2 ontology. More precisely, given an OWL 2 ontology $\mathcal{O}$, and a fragment $\mathcal{L}$ of OWL 2 , our classification algorithm proceeds as follows:

1. It computes a signature $\Sigma^{\mathcal{L}} \subseteq \mathsf{Sig}(\mathcal{O})$ and an $\mathcal{L}$-ontology $\mathcal{M}^{\mathcal{L}} \subseteq \mathcal{O}$ such that the classes in $\Sigma^{\mathcal{L}}$ can be completely classified using only the axioms in $\mathcal{M}^{\mathcal{L}}$.
2. It computes an ontology $\mathcal{M}^{\overline{\mathcal{L}}} \subseteq \mathcal{O}$ such that the classes in $\mathcal{O} \setminus \Sigma^{\mathcal{L}}$ can be fully classified using only the axioms in $\mathcal{M}^{\overline{\mathcal{L}}}$.
3. It classifies $\mathcal{M}^{\overline{\mathcal{L}}}$ using a fully-fledged OWL 2 reasoner and then completes the classification of $\mathcal{O}$ by classifying $\mathcal{M}^{\mathcal{L}}$ using an $\mathcal{L}$-reasoner.

Step 1 involves two important technical challenges. First, $\Sigma^{\mathcal{L}}$ should be as large as possible; in particular, for ontologies with only a few non $\mathcal{L}$-axioms, it is reasonable to expect $\Sigma^{\mathcal{L}}$ to contain most of the ontology's signature. Second, $\mathcal{M}^{\mathcal{L}}$ must be *complete* for $\Sigma^{\mathcal{L}}$; i.e., if a subsumption relationship between two classes in $\Sigma^{\mathcal{L}}$ is entailed by $\mathcal{O}$, then it must also be entailed by $\mathcal{M}^{\mathcal{L}}$. Although techniques such as the completely defined concepts optimisation can be used to identify a fragment of the ontology that is complete for a certain signature, these techniques are very restricted—they are not applicable to all OWL 2 ontologies, and even when they are applicable they use a fixed fragment of OWL 2 that is much smaller than the OWL 2 EL profile. In contrast, we exploit module extraction [5,4] to develop a technique that provides the following compelling features:

– It is general and flexible, as it is neither tied to any particular fragment or profile $\mathcal{L}$ of OWL 2, nor to any particular reasoner or reasoning technique.
– It is easy to implement, as reasoners are combined in a black-box manner, with no modification of their internals being required.
– It exhibits "pay-as-you-go" behaviour when an $\mathcal{L}$-ontology is extended with axioms outside $\mathcal{L}$: on the one hand, the use of an $\mathcal{L}$-reasoner is not precluded by the extension; on the other hand, performance degrades gracefully with the number of additional non $\mathcal{L}$-axioms.

We believe that our results are interesting from both a theoretical and a practical point of view. From a theoretical point of view, we show that given an OWL 2 ontology $\mathcal{O}$ that is not captured by any known tractable fragment of OWL 2, it is often possible to identify a large subset Σ of its signature such that all subsumers of classes in Σ w.r.t. $\mathcal{O}$ can be computed using a polynomial time classification algorithm. From a practical point of view, our experiments with a prototype implementation MORe,[1] which integrates the OWL 2 reasoner HermiT and the OWL 2 EL reasoner ELK, illustrate the potential of this approach for optimising classification and providing a modular reasoning system with robust pay-as-you-go performance.

2 Preliminaries

We assume basic familiarity with the W3C standard OWL 2 [6] and its EL profile [19]. When talking about *ontologies* and *axioms* we implicitly refer to OWL 2 ontologies and axioms, and when talking about OWL 2 we implicitly refer to OWL 2 under the *direct semantics* [20].

For compactness reasons, we adopt description logic notation rather than OWL syntax in examples and definitions; hence we also assume basic familiarity with the syntax and semantics of the DLs $\mathcal{SROIQ}$ [12] and $\mathcal{EL}^{++}$ [1], which provide the logical underpinning for OWL 2 and OWL 2 EL, respectively.

We consider the standard notions of signature (or vocabulary), interpretations and models, entailment, satisfiability, and class subsumption. We denote with $\mathsf{Sig}(\mathcal{O})$ (respectively, $\mathsf{Sig}(\alpha)$) the signature of an ontology $\mathcal{O}$ (respectively, of an axiom α), and use the greek letters Σ and Γ to denote signatures. We denote with $\mathcal{L}$ a generic fragment of OWL 2—either one of its profiles or any other possible fragment for which we may have an efficient reasoner. Finally, given an OWL 2 ontology $\mathcal{O}$, we denote with $\mathcal{O}_{\mathcal{L}}$ the set of $\mathcal{L}$-axioms in $\mathcal{O}$.

2.1 Module Extraction

Intuitively, a module $\mathcal{M}$ for an ontology $\mathcal{O}$ w.r.t. a signature Σ is an ontology $\mathcal{M} \subseteq \mathcal{O}$ such that $\mathcal{M}$ and $\mathcal{O}$ entail the same axioms over Σ [5].

[1] Modular OWL Reasoner.

$$\underline{\mathsf{BursitisOrCellulitisOfKnee} \equiv \mathsf{BursitisOfKnee} \sqcup \mathsf{CellulitisOfKnee}} \tag{1}$$
$$\mathsf{BursitisOfKnee} \equiv \mathsf{Bursitis} \sqcap \exists \mathsf{hasLocation}.\mathsf{Knee} \tag{2}$$
$$\mathsf{BursitisOfJoint} \equiv \mathsf{Bursitis} \sqcap \exists \mathsf{hasLocation}.\mathsf{Joint} \tag{3}$$
$$\mathsf{Bursitis} \sqsubseteq \mathsf{Swelling} \tag{4}$$
$$\mathsf{Cellulitis} \sqsubseteq \mathsf{Swelling} \sqcap \exists \mathsf{hasOrigin}.\mathsf{Infection} \tag{5}$$
$$\mathsf{InfectiousDisease} \equiv \exists \mathsf{hasOrigin}.\mathsf{Infection} \tag{6}$$
$$\exists \mathsf{hasOrigin}.\top \sqsubseteq \mathsf{Disease} \tag{7}$$
$$\mathsf{Knee} \sqsubseteq \mathsf{Joint} \tag{8}$$

Fig. 1. The example ontology $\mathcal{O}^{\mathsf{ex}}$. Its only non OWL 2 EL axiom is underlined

This intuition is typically formalised using the notions of deductive and model-based *conservative extensions* [17,5]. In this paper, we define modules in terms of the model-based notion of conservative extension.

Definition 1 (Model Conservative Extension). *Let $\mathcal{O}$ be an ontology and let $\Sigma \subseteq \mathsf{Sig}(\mathcal{O})$. We say that $\mathcal{O}$ is a* model conservative extension *of $\mathcal{M} \subseteq \mathcal{O}$ w.r.t. Σ if, for every model $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ of $\mathcal{M}$, there exists a model $\mathcal{J} = (\Delta^{\mathcal{J}}, \cdot^{\mathcal{J}})$ of $\mathcal{O}$ such that $\Delta^{\mathcal{I}} = \Delta^{\mathcal{J}}$ and $X^{\mathcal{I}} = X^{\mathcal{J}}$ for every symbol $X \in \Sigma$.*

That is, $\mathcal{O}$ is a model conservative extension of $\mathcal{M}$ for Σ if every model of $\mathcal{M}$ can be extended to a model of $\mathcal{O}$ without changing either the interpretation domain, or the interpretation of the symbols in Σ.

Definition 2 (Module). *Let $\mathcal{O}$ be an ontology and let Σ be a signature. We say that $\mathcal{M} \subseteq \mathcal{O}$ is a module for $\mathcal{O}$ w.r.t. Σ if $\mathcal{O}$ is a model conservative extension of $\mathcal{M}$ w.r.t. Σ.*

In particular, if $\mathcal{M}$ is a module for $\mathcal{O}$ w.r.t. Σ, then the following condition holds: for each axiom α with $\mathsf{Sig}(\alpha) \subseteq \Sigma$, we have $\mathcal{M} \models \alpha$ iff $\mathcal{O} \models \alpha$.

Example 1. Consider the ontology $\mathcal{O}^{\mathsf{ex}}$, given in Figure 1, which we will use as a running example. Consider also the fragment $\mathcal{O}^{\mathsf{ex}}_1 = \{(1), (2), (3), (4), (8)\}$ of $\mathcal{O}^{\mathsf{ex}}$. Let $\mathcal{I}$ be an arbitrary model of $\mathcal{O}^{\mathsf{ex}}_1$. We can obtain a model of $\mathcal{O}^{\mathsf{ex}}$ by interpreting all symbols in $\mathsf{Sig}(\mathcal{O}^{\mathsf{ex}}_1)$ in the same way as $\mathcal{I}$, and all symbols outside $\mathsf{Sig}(\mathcal{O}^{\mathsf{ex}}_1)$ as the empty set. Thus, $\mathcal{O}^{\mathsf{ex}}$ is a model conservative extension of $\mathcal{O}^{\mathsf{ex}}_1$, and $\mathcal{O}^{\mathsf{ex}}_1$ is a module for $\mathcal{O}^{\mathsf{ex}}$ w.r.t. $\mathsf{Sig}(\mathcal{O}^{\mathsf{ex}}_1)$. As a result, $\mathcal{O}^{\mathsf{ex}}$ and $\mathcal{O}^{\mathsf{ex}}_1$ entail exactly the same axioms constructed using only symbols from $\mathsf{Sig}(\mathcal{O}^{\mathsf{ex}}_1)$. ♢

The problem of checking whether $\mathcal{M}$ is a module for $\mathcal{O}$ w.r.t. Σ is, however, already undecidable for fairly lightweight fragments of OWL 2, such as the OWL 2 EL profile [18]; therefore, approximations are needed in practice. The following sufficient condition for model conservativity is known to work well [5,7].

Definition 3 ($\emptyset$-locality). *Let Σ be a signature. An interpretation $\mathcal{I}$ is $\emptyset$-local for Σ if for every class $A \notin \Sigma$ and every property $R \notin \Sigma$, we have $A^\mathcal{I} = R^\mathcal{I} = \emptyset$. An axiom α is $\emptyset$-local for Σ if $\mathcal{I} \models \alpha$ for each $\mathcal{I}$ that is $\emptyset$-local for Σ. An ontology $\mathcal{O}$ is $\emptyset$-local for Σ if every axiom in $\mathcal{O}$ is $\emptyset$-local for Σ.*

Example 2. It is easy to check that the ontology $\mathcal{O}^{\mathsf{ex}} \setminus \mathcal{O}_1^{\mathsf{ex}}$, consisting of axioms (5), (6) and (7), is $\emptyset$-local w.r.t. $\mathsf{Sig}(\mathcal{O}_1^{\mathsf{ex}})$. For example, to see that axiom (5) is indeed $\emptyset$-local w.r.t. $\mathsf{Sig}(\mathcal{O}_1^{\mathsf{ex}})$, consider any $\mathcal{I}$ that interprets all symbols in (5) other than those in $\mathsf{Sig}(\mathcal{O}_1^{\mathsf{ex}})$ as the empty set. Thus, we have $\mathsf{Cellulitis}^\mathcal{I} = \mathsf{hasOrigin}^\mathcal{I} = \mathsf{Infection}^\mathcal{I} = \emptyset$. Clearly, both left and right hand sides of axiom (5) are interpreted as the empty set by $\mathcal{I}$ (see below) and hence $\mathcal{I}$ satisfies (5).

$$\overbrace{\mathsf{Cellulitis}}^{\emptyset} \sqsubseteq \underbrace{\mathsf{Swelling} \sqcap \exists\, \overbrace{\mathsf{hasOrigin}}^{\emptyset} . \overbrace{\mathsf{Infection}}^{\emptyset}}_{\emptyset}$$ ◇

Checking $\emptyset$-locality for OWL 2 axioms is, however, a PSPACE-complete problem [5]. Since our goal is to optimise classification, checking $\emptyset$-locality might still be too costly. Instead, we will use $\bot$-locality—a well-known sufficient syntactic condition for $\emptyset$-locality which has been successfully used for both ontology reuse and reasoning problems [5,14,7,4].

The grammar defining $\bot$-locality for $\mathcal{SROIQ}$ can be found in the literature [4,5]. It suffices to note that $\bot$-locality can be checked in polynomial time and that it implies $\emptyset$-locality. Furthermore, the following property holds [4,5]:

Proposition 1. *If an axiom α is $\bot$-local w.r.t. a signature Σ, then α is $\bot$-local w.r.t. Σ' for any $\Sigma' \subseteq \Sigma$.*

We use $\bot$-locality to define the notion of $\bot$-module. The fact that $\bot$-locality implies $\emptyset$-locality ensures that, if $\mathcal{M}$ is a $\bot$-module for $\mathcal{O}$ w.r.t. Σ (as defined next), then it is a module for $\mathcal{O}$ w.r.t. Σ.

Definition 4 ($\bot$-module). *Let $\mathcal{O}$ be an ontology and let Σ be a signature. We say that $\mathcal{M} \subseteq \mathcal{O}$ is a $\bot$-module for $\mathcal{O}$ w.r.t. Σ if $\mathcal{O} \setminus \mathcal{M}$ is $\bot$-local for $\Sigma \cup \mathsf{Sig}(\mathcal{M})$.*

Example 3. It was pointed out in Example 2 that $\mathcal{O}^{\mathsf{ex}} \setminus \mathcal{O}_1^{\mathsf{ex}}$ is $\emptyset$-local w.r.t. $\mathsf{Sig}(\mathcal{O}_1^{\mathsf{ex}})$. In particular, $\mathcal{O}^{\mathsf{ex}} \setminus \mathcal{O}_1^{\mathsf{ex}}$ is also $\bot$-local w.r.t $\mathsf{Sig}(\mathcal{O}_1^{\mathsf{ex}})$, and therefore $\mathcal{O}_1^{\mathsf{ex}}$ is a $\bot$-module for $\mathcal{O}^{\mathsf{ex}}$ w.r.t $\mathsf{Sig}(\mathcal{O}_1^{\mathsf{ex}})$. ◇

The algorithm for $\bot$-module extraction [7] is given in Algorithm 1. This algorithm computes the unique smallest $\bot$-module for a given $\mathcal{O}$ and Σ (the smallest subset $\mathcal{M} \subseteq \mathcal{O}$ s.t. $\mathcal{O} \setminus \mathcal{M}$ is $\bot$-local for $\Sigma \cup \mathsf{Sig}(\mathcal{M})$). We refer to such a smallest $\bot$-module as *the* $\bot$-module for $\mathcal{O}$ w.r.t. Σ, and denote it with $\mathcal{M}_{[\mathcal{O},\Sigma]}$.

Example 4. Suppose that we want to extract a $\bot$-module for $\mathcal{O}$ w.r.t. Γ, with

$$\Gamma = \{\mathsf{Knee}, \mathsf{Bursitis}, \mathsf{hasLocation}\}$$

It can be observed in Algorithm 1 that new symbols added to the module's signature in some iteration may cause more axioms to be added to the module

Algorithm 1. $\bot$-module$(\mathcal{O}, \Sigma)$

Input: an ontology $\mathcal{O}$ and a signature Σ

```
1: $\mathcal{M} := \emptyset$; $\mathcal{O}' := \mathcal{O}$
2: repeat
3:     changed := false
4:     for all $\alpha \in \mathcal{O}'$ do
5:         if $\alpha$ not $\bot$-local w.r.t $\Sigma \cup \mathsf{Sig}(\mathcal{M})$ then
6:             $\mathcal{M} := \mathcal{M} \cup \{\alpha\}$; $\mathcal{O}' := \mathcal{O}' \setminus \{\alpha\}$
7:             changed := true
8: until changed = false
9: return $\mathcal{M}$
```

in subsequent iterations. The algorithm stops once a fixpoint is reached and no more symbols need to be added to the module's signature.

On the first iteration, we would only add axioms (2), (4) and (8) to our module. Then, due to having added Joint and BursitisOfKnee to the module's signature, we would have to add axioms (1) and (3) as well. We would thus find that $\mathcal{M}_{[\mathcal{O}^{\mathsf{ex}},\Gamma]}$ coincides with $\mathcal{O}_1^{\mathsf{ex}}$ and its signature is precisely $\mathsf{Sig}(\mathcal{O}_1^{\mathsf{ex}})$. $\diamondsuit$

In addition to being modules as in Definition 2, $\bot$-modules enjoy a property that makes them especially well-suited for optimising classification [4].

Proposition 2. *Let $\mathcal{O}$ be an ontology, let A, B be classes in $\mathsf{Sig}(\mathcal{O}) \cup \{\top, \bot\}$, let $\Sigma \subseteq \mathsf{Sig}(\mathcal{O})$ with $A \in \Sigma$, and let $\mathcal{M} \subseteq \mathcal{O}$ be a $\bot$-module in $\mathcal{O}$ w.r.t. Σ. Then $\mathcal{O} \models A \sqsubseteq B$ iff $\mathcal{M} \models A \sqsubseteq B$.*

Example 5. Proposition 2 implies that $\mathcal{O} \not\models \mathsf{Bursitis} \sqsubseteq \mathsf{Cellulitis}$. Indeed, we have that $\mathsf{Bursitis} \in \Gamma$ but $\mathsf{Cellulitis} \notin \mathsf{Sig}(\mathcal{M}_{[\mathcal{O},\Gamma]})$; therefore, it must be the case that $\mathcal{M}_{[\mathcal{O},\Gamma]} \not\models \mathsf{Bursitis} \sqsubseteq \mathsf{Cellulitis}$, and thus $\mathcal{O} \not\models \mathsf{Bursitis} \sqsubseteq \mathsf{Cellulitis}$. $\diamondsuit$

3 Modular Classification of Ontologies

Consider an ontology $\mathcal{O}$ such that most of the axioms in it are expressed in some restricted fragment $\mathcal{L}$ of OWL 2. This is the case, considering $\mathcal{L} =$ OWL 2 EL, for our example ontology $\mathcal{O}^{\mathsf{ex}}$, whose $\mathcal{L}$-fragment $\mathcal{O}^{\mathsf{ex}}_{\mathcal{L}}$ contains all the axioms in $\mathcal{O}^{\mathsf{ex}}$ except axiom (1).

Our first goal is to identify a signature $\Sigma^{\mathcal{L}} \subseteq \mathsf{Sig}(\mathcal{O})$ such that the corresponding $\bot$-module $\mathcal{M}_{[\mathcal{O},\Sigma^{\mathcal{L}}]}$ is contained in the $\mathcal{L}$-fragment $\mathcal{O}_{\mathcal{L}}$ of $\mathcal{O}$. We call any such $\Sigma^{\mathcal{L}}$ an *$\mathcal{L}$-signature for $\mathcal{O}$*. Proposition 2 ensures that an $\mathcal{L}$-reasoner can then be used to determine all the subsumers of classes in $\Sigma^{\mathcal{L}}$. Section 3.1 addresses the problem of identifying as large an $\mathcal{L}$-signature as possible.

Once an $\mathcal{L}$-signature $\Sigma^{\mathcal{L}}$ has been identified, the use of a fully-fledged OWL 2 reasoner can be restricted to computing the subsumers of the classes in the complementary signature $\overline{\Sigma^{\mathcal{L}}} = \mathsf{Sig}(\mathcal{O}) \setminus \Sigma^{\mathcal{L}}$. The details of how our classification algorithm combines the use of both an $\mathcal{L}$-reasoner and an OWL 2 reasoner as *black boxes* are given in Section 3.2.

3.1 Computing an $\mathcal{L}$-signature

The definition of $\bot$-module suggests a simple "guess and check" algorithm for computing a (maximal) $\mathcal{L}$-signature for $\mathcal{O}$: consider all subsets $\Sigma \subseteq \mathsf{Sig}(\mathcal{O})$ in decreasing size order and, for each of them, check whether $\mathcal{M}_{[\mathcal{O},\Sigma]}$ is an $\mathcal{L}$-ontology. This could, however, be quite costly, and as our objective is to optimise classification we propose a goal directed algorithm. Although our algorithm is not guaranteed to compute a maximal $\mathcal{L}$-signature, it can be implemented very efficiently; furthermore, as shown in the evaluation section, it typically computes large $\mathcal{L}$-signatures, provided that $\mathcal{O}_{\mathcal{L}}$ is a large enough fragment of $\mathcal{O}$.

We start by pointing out that every $\mathcal{L}$-signature $\Sigma^{\mathcal{L}}$ *must* satisfy the property ($\star$) below. If ($\star$) does not hold, then $\mathcal{M}_{[\mathcal{O},\Sigma^{\mathcal{L}}]}$ will contain some non $\mathcal{L}$-axiom.

$$\text{Property } (\star): \quad \mathcal{O} \setminus \mathcal{O}_{\mathcal{L}} \text{ is } \bot\text{-local w.r.t. } \Sigma^{\mathcal{L}}$$

Example 6. Consider again our example ontology $\mathcal{O}^{\mathsf{ex}}$ and let $\mathcal{L}$ be OWL 2 EL. As already mentioned, the $\mathcal{L}$-fragment $\mathcal{O}^{\mathsf{ex}}_{\mathcal{L}}$ of $\mathcal{O}^{\mathsf{ex}}$ contains all axioms in $\mathcal{O}^{\mathsf{ex}}$ except for (1). One may think that the signature of $\mathcal{O}^{\mathsf{ex}}_{\mathcal{L}}$ is an $\mathcal{L}$-signature, which makes the computation of a maximal $\mathcal{L}$-signature trivial; this is, however, not the case. Note that the signature of $\mathcal{O}^{\mathsf{ex}}_{\mathcal{L}}$, namely

$$\mathsf{Sig}(\mathcal{O}^{\mathsf{ex}}_{\mathcal{L}}) = \mathsf{Sig}(\mathcal{O}^{\mathsf{ex}}) \setminus \{\mathsf{BursitisOrCellulitisOfKnee}, \mathsf{CellulitisOfKnee}\}$$

is not an $\mathcal{L}$-signature for $\mathcal{O}^{\mathsf{ex}}$; indeed, axiom (1) is not $\bot$-local w.r.t $\mathsf{Sig}(\mathcal{O}^{\mathsf{ex}}_{\mathcal{L}})$. In contrast, we have that axiom (1) is $\bot$-local w.r.t.

$$\begin{aligned}\Gamma' = \{&\mathsf{Bursitis}, \mathsf{hasLocation}, \mathsf{Joint}, \mathsf{BursitisOfJoint}, \mathsf{Swelling}, \mathsf{Infection},\\ &\mathsf{InfectiousDisease}, \mathsf{Disease}, \mathsf{hasOrigin}\}\end{aligned}$$

Furthermore, $\mathcal{M}_{[\mathcal{O}^{\mathsf{ex}},\Gamma']}$ consists of axioms (3), (4) (5) (6) (7), which are all within the OWL 2 EL fragment; hence, Γ' is an $\mathcal{L}$-signature for $\mathcal{O}^{\mathsf{ex}}$. ♢

Although Example 6 might suggest that property ($\star$) is also a *sufficient* condition for $\Sigma^{\mathcal{L}}$ to be an $\mathcal{L}$-signature in $\mathcal{O}$, this is unfortunately not the case.

Example 7. Consider again the signature Γ from Example 4. Clearly, axiom (1) (the only non $\mathcal{L}$-axiom in $\mathcal{O}^{\mathsf{ex}}$) is $\bot$-local w.r.t Γ and hence ($\star$) holds for Γ. Note, however, that Γ is not an $\mathcal{L}$-signature for $\mathcal{O}^{\mathsf{ex}}$ since, as already discussed, axiom (1) is contained in $\mathcal{M}_{[\mathcal{O}^{\mathsf{ex}},\Gamma]}$. One way to address this problem is to reduce Γ to $\Gamma \setminus \{\mathsf{Knee}\}$. The corresponding $\bot$-module only contains axiom (4), which implies that such reduced signature is indeed an $\mathcal{L}$-signature for $\mathcal{O}^{\mathsf{ex}}$. ♢

Example 7 suggests an algorithm for computing an $\mathcal{L}$-signature for $\mathcal{O}$, which can be intuitively described as follows.

1. Reduce $\Sigma_0 = \mathsf{Sig}(\mathcal{O})$ to a subset Σ_1 of Σ_0 such that $\mathcal{S}_0 = \mathcal{O} \setminus \mathcal{O}_{\mathcal{L}}$ is $\bot$-local w.r.t. Σ_1 (thus satisfying $(\star)$).
2. Compute the set $\mathcal{S}_1$ of axioms in $\mathcal{M}_{[\mathcal{O},\Sigma_1]}$ containing symbols not in Σ_1.
3. Reduce Σ_1 to a subset Σ_2 of Σ_1 such that $\mathcal{S}_1$ is $\bot$-local w.r.t. Σ_2.
4. Repeat Steps [2-4] until the set of axioms computed in Step 2 is empty.

The sequence $(\Sigma_i)_{i\geq 0}$ will eventually converge to a fixpoint (as we will shortly prove), and this fixpoint will be guaranteed to be an $\mathcal{L}$-signature, $\Sigma^{\mathcal{L}}$. We next explain the intuition behind our algorithm with an example.

Example 8. Consider once more our example ontology $\mathcal{O}^{\mathsf{ex}}$. As already mentioned, the only non OWL 2 EL axiom is (1), so we start with

$$\begin{aligned}\Sigma_0 &= \mathsf{Sig}(\mathcal{O}^{\mathsf{ex}})\\ \mathcal{S}_0 &= \{\mathsf{BursitisOrCellulitisOfKnee} \equiv \mathsf{BursitisOfKnee} \sqcup \mathsf{CellulitisOfKnee}\}\end{aligned}$$

The only way to make axiom (1) $\bot$-local is by removing BursitisOrCellulitisOfKnee, BursitisOfKnee and CellulitisOfKnee from Σ_0. So we take

$$\Sigma_1 = \Sigma_0 \setminus \{\mathsf{BursitisOrCellulitisOfKnee}, \mathsf{BursitisOfKnee}, \mathsf{CellulitisOfKnee}\}$$

Next, we compute $\mathcal{M}_{[\mathcal{O}^{\mathsf{ex}},\Sigma_1]}$ using Algorithm 1. This module contains axiom (2), which mentions BursitisOfKnee (not in Σ_1). Because this class is in the module's signature, the module needs to contain axiom (1) as well. This gives us

$$\begin{aligned}\mathcal{S}_1 = \{&\mathsf{BursitisOrCellulitisOfKnee} \equiv \mathsf{BursitisOfKnee} \sqcup \mathsf{CellulitisOfKnee},\\ &\mathsf{BursitisOfKnee} \equiv \mathsf{Bursitis} \sqcap \exists \mathsf{hasLocation}.\mathsf{Knee}\}\end{aligned}$$

We have seen that, unless axiom (2) in $\mathcal{S}_1$ is outside the module, axiom (1) cannot be outside the module either. Thus, we need to make sure that axiom (2) is $\bot$-local. For this, we can take $\Sigma_2 = \Sigma_1 \setminus \{\mathsf{Knee}\}$.

At this point, we need not worry about axiom (1) anymore; it was already local w.r.t. Σ_1, so, by Proposition 1, it will be $\bot$-local w.r.t. any subset of Σ_1.

Next, we compute $\mathcal{M}_{[\mathcal{O}^{\mathsf{ex}},\Sigma_2]}$ and find that it contains all axioms in $\mathcal{O}^{\mathsf{ex}}$ except for axioms (1), (2) and (8). This implies that all symbols in $\mathcal{M}_{[\mathcal{O}^{\mathsf{ex}},\Sigma_2]}$ are in Σ_2 and hence $\mathcal{S}_3 = \emptyset$. The algorithm then terminates and returns $\Sigma^{\mathcal{L}} = \Sigma_2$.

Note that $\Sigma^{\mathcal{L}}$ is indeed an $\mathcal{L}$-signature since the module $\mathcal{M}_{[\mathcal{O}^{\mathsf{ex}},\Sigma_2]}$ does not contain axiom (1) and hence is an OWL 2 EL ontology. ♢

Note that there can be many ways to perform the signature reduction required in Steps 1 and 4. In Example 8, for instance, we could have taken $\Sigma_2 = \Sigma_1 \setminus \{\mathsf{Bursitis}\}$ or $\Sigma_2 = \Sigma_1 \setminus \{\mathsf{hasLocation}\}$, or even any subset thereof. Making reasonable choices requires good heuristics. In our implementation our choices are guided so that as many properties as possible are kept within $\Sigma^{\mathcal{L}}$. Indeed, ontologies typically contain many more classes than properties, and each property typically occurs in a larger number of axioms; thus, having a property outside $\Sigma^{\mathcal{L}}$ is likely to cause many other symbols to be left outside $\Sigma^{\mathcal{L}}$. The following example illustrates how different choices lead to rather different $\mathcal{L}$-signatures.

Algorithm 2. $\mathcal{L}$-signature($\mathcal{O}$)

Input: an ontology $\mathcal{O}$

```
1: $\Sigma^{\mathcal{L}} := \mathsf{Sig}(\mathcal{O})$ ; $\mathcal{S} := \mathcal{O} \setminus \mathcal{O}_{\mathcal{L}}$
2: canReduce := **true**
3: **while** $\mathcal{S} \neq \emptyset$ and canReduce **do**
4:     $\Sigma^{\mathcal{L}} := \mathsf{reduce}(\Sigma^{\mathcal{L}}, \mathcal{S})$
5:     **if** $\Sigma^{\mathcal{L}} = \emptyset$ **then** canReduce := **false**
6:     **else** $\mathcal{S} := \{\alpha \in \mathcal{M}_{[\mathcal{O},\Sigma^{\mathcal{L}}]} \mid \mathsf{Sig}(\alpha) \not\subseteq \Sigma^{\mathcal{L}}\}$
7: **return** $\Sigma^{\mathcal{L}}$
```

Example 9. As already mentioned, in Example 8 we could have alternatively chosen to take $\Sigma_2 = \Sigma_1 \setminus \{\mathsf{hasLocation}\}$. This would have caused $\mathcal{M}_{[\mathcal{O},\Sigma_2]} = \mathcal{O}^{\mathsf{ex}}$ again, and we would have obtained

$$\mathcal{S}_3 = \mathcal{S}_2 \cup \{\mathsf{BursitisOfJoint} \equiv \mathsf{Bursitis} \sqcap \exists \mathsf{hasLocation.Joint}\}$$

It would now be enough to take $\Sigma_3 = \Sigma_2 \setminus \{\mathsf{BursitisOfJoint}\}$, and we would have $\mathcal{S}_4 = \emptyset$. In this case we would get $\Sigma^{\mathcal{L}} = \Sigma_3$, which is smaller (and hence less appealing) than the $\Sigma^{\mathcal{L}}$ obtained in Example 8. ♢

These signature reductions satisfy the same properties that make them "acceptable". We can characterise these acceptable reductions as given next.

Definition 5. *Given an ontology $\mathcal{O}$, a signature reduction is a function*

$$\mathsf{reduce} : \mathcal{P}(\mathsf{Sig}(\mathcal{O})) \times \mathcal{P}(\mathcal{O}) \to \mathcal{P}(\mathsf{Sig}(\mathcal{O}))$$

that, given $\Sigma \in \mathcal{P}(\mathsf{Sig}(\mathcal{O}))$ and $\mathcal{S} \in \mathcal{P}(\mathcal{O})$ not $\bot$-local w.r.t. Σ, returns

1. *Σ if $\mathcal{S} = \emptyset$.*
2. *$\Sigma' \subset \Sigma$ s.t. each axiom in $\mathcal{S}$ is $\bot$-local w.r.t. Σ' if $\mathcal{S} \neq \emptyset$ and Σ' exists.*
3. *$\emptyset$ otherwise.*

Note that the Cases 1 and 3 correspond to the extreme situations when $\mathcal{S} = \emptyset$ or when there is no satisfying way of reducing Σ. Case 2 constitutes the essence of the reduction, namely to compute a strict subset of the signature that makes the given set of axioms $\bot$-local.

Given a particular reduce function, Algorithm 2 accepts an ontology $\mathcal{O}$ and returns a signature $\Sigma^{\mathcal{L}} \subseteq \mathsf{Sig}(\mathcal{O})$. Theorem 1 guarantees the termination of Algorithm 2 as well as its correctness: any non-empty signature returned by the algorithm is an $\mathcal{L}$-signature for $\mathcal{O}$.

Theorem 1. *Let $\mathcal{O}$ be an ontology and let* reduce *be a signature reduction function. Furthermore, let $\mathcal{S}_i$, Σ_i ($i \geq 0$) be defined by the following construction:*

$$\begin{array}{lll} (i = 0): & \Sigma_0 = \mathsf{Sig}(\mathcal{O}) & \mathcal{S}_0 = \mathcal{O} \setminus \mathcal{O}_{\mathcal{L}} \\ (i \geq 1): & \Sigma_i = \mathsf{reduce}(\Sigma_{i-1}, \mathcal{S}_{i-1}) & \mathcal{S}_i = \{\alpha \in \mathcal{M}_{[\mathcal{O},\Sigma_i]} \mid \mathsf{Sig}(\alpha) \not\subseteq \Sigma_i\} \end{array}$$

Finally, let $\Sigma^{\mathcal{L}} := \bigcap_{i \geq 0} \Sigma_i$. *Then, the following properties hold:*

1. *There exists* $k < |\mathsf{Sig}(\mathcal{O})|$ *such that either* $\Sigma_k = \emptyset$ *or* $\mathcal{S}_k = \emptyset$.
2. *Either* $\Sigma^{\mathcal{L}} = \emptyset$ *or* $\mathcal{M}_{[\mathcal{O},\Sigma^{\mathcal{L}}]} \subseteq \mathcal{O}_{\mathcal{L}}$.

Proof. We first show Claim 1. Suppose $\Sigma_i \neq \emptyset$ for each $i \geq 0$. A straightforward inductive argument would show that $\Sigma_j \subseteq \Sigma_i$ for each $j > i \geq 0$. Furthermore, $\Sigma_0 = \mathsf{Sig}(\mathcal{O})$, so it cannot be the case that $\Sigma_j \subset \Sigma_i$ for each $0 \leq i < j \leq |\mathsf{Sig}(\mathcal{O})|$. Therefore, there must be some $k < |\mathsf{Sig}(\mathcal{O})|$ such that $\Sigma_{k+1} = \Sigma_k$; by the definition of reduce, this implies that $\mathcal{S}_k = \emptyset$.

We finally show Claim 2. Suppose $\Sigma^{\mathcal{L}} \neq \emptyset$. It is enough to prove that each $\alpha \in \mathcal{O} \setminus \mathcal{O}_{\mathcal{L}}$ is $\bot$-local w.r.t. $\Sigma^{\mathcal{L}} \cup \mathsf{Sig}(\mathcal{M}_{[\mathcal{O},\Sigma^{\mathcal{L}}]})$.

First, we are going to see that $\mathsf{Sig}(\mathcal{M}_{[\mathcal{O},\Sigma^{\mathcal{L}}]}) \subseteq \Sigma^{\mathcal{L}}$. According to Claim 1, there exists $k < |\mathsf{Sig}(\mathcal{O})|$ such that $\mathcal{S}_k = \emptyset$. This implies that, for each axiom $\alpha \in \mathcal{M}_{[\mathcal{O},\Sigma_k]}$, we have $\mathsf{Sig}(\alpha) \subseteq \Sigma_k$. It is easy to see that $\mathcal{S}_k = \emptyset$ also implies that $\Sigma_j = \Sigma_k$ for each $j > k$. Together with the fact that $\Sigma_j \subseteq \Sigma_i$ for each $j > i \geq 0$, this implies $\Sigma^{\mathcal{L}} = \bigcap_{i \geq 0} \Sigma_i = \Sigma_k$. But then for each $\alpha \in \mathcal{M}_{[\mathcal{O},\Sigma^{\mathcal{L}}]} = \mathcal{M}_{[\mathcal{O},\Sigma_k]}$ we have $\mathsf{Sig}(\alpha) \subseteq \Sigma_k = \Sigma^{\mathcal{L}}$, and so $\mathsf{Sig}(\mathcal{M}_{[\mathcal{O},\Sigma^{\mathcal{L}}]}) \subseteq \Sigma^{\mathcal{L}}$.

Now we can just prove that each $\alpha \in \mathcal{O} \setminus \mathcal{O}_{\mathcal{L}}$ is $\bot$-local w.r.t. $\Sigma^{\mathcal{L}}$. Because $\Sigma^{\mathcal{L}} = \bigcap_{i \geq 0} \Sigma_i \neq \emptyset$, in particular it must be the case that $\Sigma_0 \neq \emptyset$. By definition of reduce, either $\mathcal{O} \setminus \mathcal{O}_{\mathcal{L}} = \emptyset$, in which case it is immediate that $\mathcal{M}_{[\mathcal{O},\Sigma^{\mathcal{L}}]} \subseteq \mathcal{O}_{\mathcal{L}}$, or every axiom in $\mathcal{S}_0 = \mathcal{O} \setminus \mathcal{O}_{\mathcal{L}}$ is $\bot$-local w.r.t. $\Sigma_1 = \mathsf{reduce}(\Sigma_0, \mathcal{S}_0)$. Then, by Proposition 1, each $\alpha \in \mathcal{O} \setminus \mathcal{O}_{\mathcal{L}}$ is $\bot$-local w.r.t. $\Sigma^{\mathcal{L}} \subseteq \Sigma_1$. □

3.2 Black-Box Modular Classification

Having identified a (hopefully large) $\mathcal{L}$-signature for our ontology $\mathcal{O}$, we can next proceed to classify the ontology in a modular way.

As already mentioned, we can fully classify the classes in $\Sigma^{\mathcal{L}}$ using only an $\mathcal{L}$-reasoner. This is a consequence of Proposition 2 and the fact that $\mathcal{M}_{[\mathcal{O},\Sigma^{\mathcal{L}}]} \subseteq \mathcal{O}_{\mathcal{L}}$.

To classify the classes in $\overline{\Sigma^{\mathcal{L}}} = \mathsf{Sig}(\mathcal{O}) \setminus \Sigma^{\mathcal{L}}$, however, a fully fledged OWL 2 reasoner is still required. By Proposition 2, the OWL 2 reasoner does not need to consider all the axioms in $\mathcal{O}$, but only those in the relevant module $\mathcal{M}_{[\mathcal{O},\overline{\Sigma^{\mathcal{L}}}]}$.

Once the OWL 2 reasoner has computed the classification of $\mathcal{M}_{[\mathcal{O},\overline{\Sigma^{\mathcal{L}}}]}$, we can express the classification result as simple subsumption axioms of the form $A \sqsubseteq B$. These axioms, together with $\mathcal{M}_{[\mathcal{O},\Sigma^{\mathcal{L}}]}$, can be given to the $\mathcal{L}$-reasoner, which will use the resulting ontology to compute a complete classification of $\mathcal{O}$.

Algorithm 3 describes the entire classification process for a given fragment $\mathcal{L}$ of OWL 2 and a particular signature reduction function reduce. The function $\mathcal{L}$-signature is as given in Algorithm 2. The function classification returns the classification of the given ontology (computed using an OWL 2 reasoner) as a set of axioms of the form $A \sqsubseteq B$. The function $\mathsf{classification}_{\mathcal{L}}$ returns the classification of the given ontology as computed by an $\mathcal{L}$-reasoner.

```
Algorithm 3. $\mathcal{L}$-ModularClassification($\mathcal{O}$)
Input: an OWL 2 ontology $\mathcal{O}$
1: $\Sigma^{\mathcal{L}} := \mathcal{L}\text{-signature}(\mathcal{O})$                          ▷ See Algorithm 2
2: $\mathcal{H}_{\overline{\Sigma^{\mathcal{L}}}} := \text{classification}(\mathcal{M}_{[\mathcal{O},\overline{\Sigma^{\mathcal{L}}}]})$         ▷ using the OWL 2 reasoner
3: $\mathcal{H} := \text{classification}_{\mathcal{L}}(\mathcal{M}_{[\mathcal{O},\Sigma^{\mathcal{L}}]} \cup \mathcal{H}_{\overline{\Sigma^{\mathcal{L}}}})$      ▷ using the $\mathcal{L}$-reasoner
4: return $\mathcal{H}$
```

Example 10. Recall the $\mathcal{L}$-signature $\Sigma^{\mathcal{L}}$ for $\mathcal{O}^{\mathsf{ex}}$ computed in Example 8:

$$\Sigma^{\mathcal{L}} = \{\mathsf{Bursitis}, \mathsf{hasLocation}, \mathsf{BursitisOfJoint}, \mathsf{Joint}, \mathsf{Swelling}, \mathsf{Cellulitis}, \mathsf{hasOrigin}, \mathsf{Infection}, \mathsf{InfectiousDisease}, \mathsf{Disease}\}$$

The complementary signature is

$$\overline{\Sigma^{\mathcal{L}}} = \{\mathsf{BursitisOrCellulitisOfKnee}, \mathsf{BursitisOfKnee}, \mathsf{CellulitisOfKnee}, \mathsf{Knee}\}$$

and the relevant $\bot$-module is $\mathcal{M}_{[\mathcal{O}^{\mathsf{ex}},\overline{\Sigma^{\mathcal{L}}}]} = \{(1), (2), (3), (4), (8)\}$. The classification of this module leads to the following subsumptions $\mathcal{H}_{\overline{\Sigma^{\mathcal{L}}}}$:

$$\begin{array}{rlrl}
\mathsf{CellulitisOfKnee} \sqsubseteq \mathsf{BursitisOrCellulitisOfKnee} & & \mathsf{BursitisOfJoint} \sqsubseteq \mathsf{Bursitis} \\
\mathsf{BursitisOfKnee} \sqsubseteq \mathsf{BursitisOrCellulitisOfKnee} & & \mathsf{Bursitis} \sqsubseteq \mathsf{Swelling} \\
\mathsf{BursitisOfKnee} \sqsubseteq \mathsf{BursitisOfJoint} & & \mathsf{Knee} \sqsubseteq \mathsf{Joint} \\
\mathsf{BursitisOfKnee} \sqsubseteq \mathsf{Bursitis} & &
\end{array}$$

We can now use the $\mathcal{L}$-reasoner to classify $\mathcal{M}_{[\mathcal{O},\Sigma^{\mathcal{L}}]} \cup \mathcal{H}_{\overline{\Sigma^{\mathcal{L}}}}$, thus obtaining all the remaining subsumption relationships that hold in $\mathcal{O}^{\mathsf{ex}}$:

$$\begin{array}{rr}
\mathsf{Cellulitis} \sqsubseteq \mathsf{Swelling} & \mathsf{Cellulitis} \sqsubseteq \mathsf{InfectiousDisease} \\
\mathsf{InfectiousDisease} \sqsubseteq \mathsf{Disease} & \mathsf{Cellulitis} \sqsubseteq \mathsf{Disease}
\end{array}$$
◇

The following Theorem establishes the correctness of Algorithm 3.

Theorem 2. *Let $\mathcal{O}$ be an ontology, let* reduce *be a signature reduction function and let $\mathcal{M}_{[\mathcal{O},\Sigma^{\mathcal{L}}]}$, $\mathcal{H}_{\overline{\Sigma^{\mathcal{L}}}}$ be as computed by Algorithm 3. Then, for any two classes $A \in \mathsf{Sig}(\mathcal{O})$ and $B \in \mathsf{Sig}(\mathcal{O}) \cup \{\top, \bot\}$, we have that*

$$\mathcal{O} \models A \sqsubseteq B \textit{ iff } (\mathcal{M}_{[\mathcal{O},\Sigma^{\mathcal{L}}]} \cup \mathcal{H}_{\overline{\Sigma^{\mathcal{L}}}}) \models A \sqsubseteq B$$

Proof. Let $A \in \mathsf{Sig}(\mathcal{O})$ and $B \in \mathsf{Sig}(\mathcal{O}) \cup \{\top, \bot\}$. We consider two cases.

– <u>Case 1</u>: $A \in \Sigma^{\mathcal{L}}$. Then, by Proposition 2, we have that $\mathcal{O} \models A \sqsubseteq B$ iff $\mathcal{M}_{[\mathcal{O},\Sigma^{\mathcal{L}}]} \models A \sqsubseteq B$. Also, because $\mathcal{M}_{[\mathcal{O},\Sigma^{\mathcal{L}}]} \subseteq \mathcal{M}_{[\mathcal{O},\Sigma^{\mathcal{L}}]} \cup \mathcal{H}_{\overline{\Sigma^{\mathcal{L}}}}$, by monotonicity we have that $\mathcal{M}_{[\mathcal{O},\Sigma^{\mathcal{L}}]} \models A \sqsubseteq B$ implies $(\mathcal{M}_{[\mathcal{O},\Sigma^{\mathcal{L}}]} \cup \mathcal{H}_{\overline{\Sigma^{\mathcal{L}}}}) \models A \sqsubseteq B$. It remains to show that

$$(\mathcal{M}_{[\mathcal{O},\Sigma^{\mathcal{L}}]} \cup \mathcal{H}_{\overline{\Sigma^{\mathcal{L}}}}) \models A \sqsubseteq B \text{ implies } \mathcal{M}_{[\mathcal{O},\Sigma^{\mathcal{L}}]} \models A \sqsubseteq B \qquad (9)$$

Because $\mathcal{H}_{\overline{\Sigma^{\mathcal{L}}}}$ encodes the classification of $\mathcal{M}_{[\mathcal{O},\overline{\Sigma^{\mathcal{L}}}]}$, we have that

$$(\mathcal{M}_{[\mathcal{O},\Sigma^{\mathcal{L}}]} \cup \mathcal{H}_{\overline{\Sigma^{\mathcal{L}}}}) \models A \sqsubseteq B \text{ implies } (\mathcal{M}_{[\mathcal{O},\Sigma^{\mathcal{L}}]} \cup \mathcal{M}_{[\mathcal{O},\overline{\Sigma^{\mathcal{L}}}]}) \models A \sqsubseteq B \quad (10)$$

Now, it is immediate that $\mathcal{M}_{[\mathcal{O},\Sigma^{\mathcal{L}}]} \cup \mathcal{M}_{[\mathcal{O},\overline{\Sigma^{\mathcal{L}}}]} \subseteq \mathcal{O}$; thus, because $\mathcal{M}_{[\mathcal{O},\Sigma^{\mathcal{L}}]}$ is a module for $\mathcal{O}$ w.r.t. $\Sigma^{\mathcal{L}}$, it must also be a module for $\mathcal{M}_{[\mathcal{O},\Sigma^{\mathcal{L}}]} \cup \mathcal{M}_{[\mathcal{O},\overline{\Sigma^{\mathcal{L}}}]}$ w.r.t. $\Sigma^{\mathcal{L}}$. So, again by Proposition 2, we have

$$(\mathcal{M}_{[\mathcal{O},\Sigma^{\mathcal{L}}]} \cup \mathcal{M}_{[\mathcal{O},\overline{\Sigma^{\mathcal{L}}}]}) \models A \sqsubseteq B \text{ implies } \mathcal{M}_{[\mathcal{O},\Sigma^{\mathcal{L}}]} \models A \sqsubseteq B \quad (11)$$

Now, (10) and (11) imply (9), as required.

– Case 2: $A \in \overline{\Sigma^{\mathcal{L}}} = \mathsf{Sig}(\mathcal{O}) \setminus \Sigma^{\mathcal{L}}$. Then, by Proposition 2, $\mathcal{O} \models A \sqsubseteq B$ iff $\mathcal{M}_{[\mathcal{O},\overline{\Sigma^{\mathcal{L}}}]} \models A \sqsubseteq B$. Because $\mathcal{H}_{\overline{\Sigma^{\mathcal{L}}}}$ represents the classification of $\mathcal{M}_{[\mathcal{O},\overline{\Sigma^{\mathcal{L}}}]}$, we have $\mathcal{M}_{[\mathcal{O},\overline{\Sigma^{\mathcal{L}}}]} \models A \sqsubseteq B$ iff $\mathcal{H}_{\overline{\Sigma^{\mathcal{L}}}} \models A \sqsubseteq B$. It remains to show that

$$\mathcal{H}_{\overline{\Sigma^{\mathcal{L}}}} \models A \sqsubseteq B \text{ iff } \mathcal{M}_{[\mathcal{O},\Sigma^{\mathcal{L}}]} \cup \mathcal{H}_{\overline{\Sigma^{\mathcal{L}}}} \models A \sqsubseteq B \quad (12)$$

Left to right implication holds directly by monotonicity, so let us assume $\mathcal{M}_{[\mathcal{O},\Sigma^{\mathcal{L}}]} \cup \mathcal{H}_{\overline{\Sigma^{\mathcal{L}}}} \models A \sqsubseteq B$. Since $\mathcal{H}_{\overline{\Sigma^{\mathcal{L}}}}$ is the classification of $\mathcal{M}_{[\mathcal{O},\overline{\Sigma^{\mathcal{L}}}]}$, we have $\mathcal{M}_{[\mathcal{O},\Sigma^{\mathcal{L}}]} \cup \mathcal{M}_{[\mathcal{O},\overline{\Sigma^{\mathcal{L}}}]} \models A \sqsubseteq B$. Now, $\mathcal{M}_{[\mathcal{O},\overline{\Sigma^{\mathcal{L}}}]}$ is a $\bot$-module for $\mathcal{M}_{[\mathcal{O},\Sigma^{\mathcal{L}}]} \cup \mathcal{M}_{[\mathcal{O},\overline{\Sigma^{\mathcal{L}}}]} \subseteq \mathcal{O}$ w.r.t. $\overline{\Sigma^{\mathcal{L}}}$ since it is a $\bot$-module for $\mathcal{O}$ w.r.t. $\overline{\Sigma^{\mathcal{L}}}$. By Proposition 2, $\mathcal{M}_{[\mathcal{O},\overline{\Sigma^{\mathcal{L}}}]} \models A \sqsubseteq B$, which implies $\mathcal{H}_{\overline{\Sigma^{\mathcal{L}}}} \models A \sqsubseteq B$. □

4 Implementation and Experiments

We have implemented our modular reasoner MORe[2] in Java using the OWL API.[3] Our implementation of a signature reduction function reduce (see Section 3.1) is based on the $\bot$-locality module extractor described in [14].[4] Our system currently integrates ELK, which acts as an OWL 2 EL reasoner, and HermiT, which plays the role of a fully-fledged OWL 2 reasoner.

In the implementation of the signature reduction, symbols required to make a set of axioms $\bot$-local are selected greedily axiom by axiom. As discussed in Examples 8 and 9, when selecting symbols it is often a good idea to implement heuristics that try to keep as many properties as possible within $\Sigma^{\mathcal{L}}$.

Evaluation on BioPortal Ontologies. We have compared classification times obtained by MORe and HermiT over a set of large bio-medical ontologies available from BioPortal.[5] Results are summarised in the upper part of Table 1. The Gene Ontology (GO) and Gazetteer are OWL 2 EL ontologies; therefore, MORe delegates all the work to ELK, with the consequent performance improvement. For the latest version of NCI and for Protein, which contain only a small number

[2] http://www.cs.ox.ac.uk/isg/tools/MORe/
[3] http://owlapi.sourceforge.net/
[4] http://www.cs.ox.ac.uk/isg/tools/ModuleExtractor
[5] http://bioportal.bioontology.org/

Table 1. MORe vs HermiT. Comparison on BioPortal ontologies and on mapped ontologies. Tables show number of axioms outside OWL 2 EL, relative size of $\Sigma^{\mathcal{L}}$ and $\mathcal{M}_{[\mathcal{O},\overline{\Sigma^{\mathcal{L}}}]}$, and classification times using HermiT and MORe. For MORe we specify the performance gain w.r.t. HermiT alone and the time taken by HermiT and ELK.

Ontology	$\lvert\mathcal{O} \setminus \mathcal{O}_{\mathcal{L}}\rvert$	$\lvert\Sigma^{\mathcal{L}}\rvert$	$\lvert\mathcal{M}_{[\mathcal{O},\overline{\Sigma^{\mathcal{L}}}]}\rvert$	**Classif. time** (seconds)			
				HermiT	MORe		
					total	HermiT	ELK
GO	0	100%	0%	7.1	2.2 (↓69.0%)	0	0.1
Gazeteer	0	100%	0%	838.1	28.2 (↓96.6%)	0	15.6
NCI	65	94.9%	15.4%	84.1	28.6 (↓66.0%)	15.8	3.3
Protein	12	98.1%	6.6%	11.4	2.9 (↓74.6%)	0.4	0.9
Biomodels	22,079	45.2%	66.4%	741.4	575.6 (↓22.4%)	540.1	2.6
cellCycle	1	$> 99.9\%$	$< 0.1\%$	–	13.9 (–)	<0.1	4.9
NCI+CHEBI	65	95.6%	10.3%	116.6	34.0 (↓70.8%)	16.3	4.1
NCI+GO	65	96.7%	10.4%	110.0	37.6 (↓65.8%)	17.6	3.2
NCI+Mouse	65	96.0%	13.3%	93.7	31.0 (↓66.9%)	16.6	2.6

of axioms outside OWL 2 EL, the obtained $\Sigma^{\mathcal{L}}$ contains most of $\mathsf{Sig}(\mathcal{O})$, and hence MORe significantly outperforms HermiT. Biomodels, however, contains a large number of axioms outside OWL 2 EL, thus the size of $\Sigma^{\mathcal{L}}$ is proportionally much smaller and MORe must assign a higher workload to HermiT, which results in a more modest performance gain. Finally, the Cell Cycle ontology is an extreme case: an ontology that is *almost* OWL 2 EL and can be classified efficiently with MORe, while HermiT alone runs out of memory.

Evaluation on Ontologies Integrated via Mappings. We have used the ontology matching tool LogMap [13] to integrate the latest version of NCI with other widely used ontologies. Results are summarised in the lower part of Table 1. We can observe that MORe consistently outperforms HermiT by 65-70%.

Evaluation on versions of NCI. We have compared MORe with HermiT on 10 versions of NCI.[6] Unsurprisingly, there have been significant variations in 10 years of development; for example, a 2003 version was entirely in OWL 2 EL, a version in 2009 contained more than 4,000 axioms outside OWL 2 EL, and the current version only contains 65. Figure 2 summarises our results; in all cases, MORe outperforms HermiT.

Extensions of SNOMED. We have manually extended SNOMED (v. Jan 2010), which is fully expressed in OWL 2 EL, with a few disjunctive axioms suggested by domain experts who are involved in SNOMED's development. All these axioms share the same structure; for example,

$$\mathsf{Sprain\_of\_ankle\_OR\_foot} \equiv \mathsf{Sprain\_of\_ankle} \sqcup \mathsf{Sprain\_of\_foot}$$

[6] See http://ncit.nci.nih.gov/. We consider the latest version in each year.

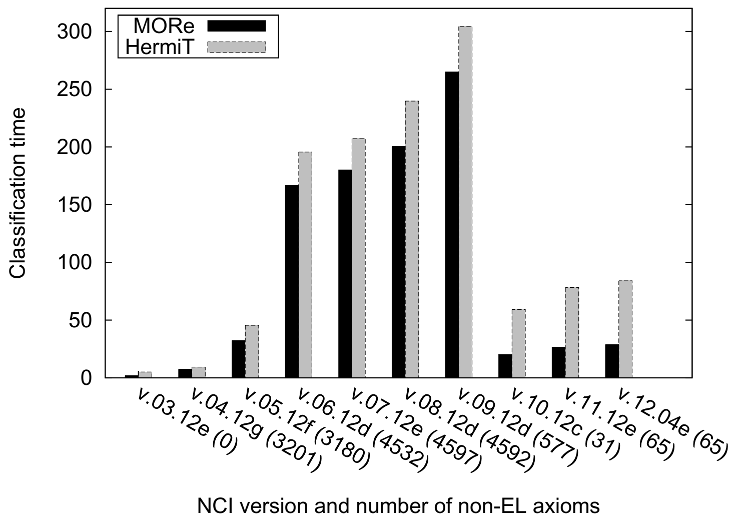


Fig. 2. Classification times (seconds) for MORe and HermiT on NCI. The X axis indicates the version and the number of axioms outside OWL 2 EL (in parenthesis).

Table 2. Extensions of SNOMED

$\#\sqcup$	$\|\Sigma^{\mathcal{L}}\|$	$\|\mathcal{M}_{[\mathcal{O},\overline{\Sigma^{\mathcal{L}}}]}\|$	Classif. time	
			HermiT	MORe
1	99.98%	0.10%	1,788.5	25.3
2	99.94%	0.24%	1,959.2	29.0
3	99.88%	0.52%	1,872.8	29.3
4	99.86%	0.61%	1,933.2	30.9
5	99.86%	0.63%	1,898.6	31.6
6	99.86%	0.63%	1,920.2	31.0
7	99.86%	0.64%	1,884.8	31.8
8	99.85%	0.65%	1,868.2	31.3
9	99.85%	0.66%	1,937.2	31.9
10	99.79%	1.00%	1,863.7	32.8
11	99.79%	1.00%	1,922.5	30.6
12	99.79%	1.01%	1,912.4	30.6
13	99.78%	1.02%	1,864.0	30.5
14	99.76%	1.91%	1,890.5	33.0
15	98.76%	3.19%	1,925.9	42.9
16	97.10%	9.79%	1,930.2	138.5
17	97.08%	9.89%	1,927.9	134.7
18	96.27%	13.50%	1,881.4	269.8
19	94.10%	17.65%	1,847.4	401.1
20	94.02%	17.78%	1,904.0	410.8

introduces a new class that is fully defined as the set of all sprains that affect either the ankle or the foot (or both). In total, 20 such axioms were added to SNOMED, one by one.

Table 2 presents the results obtained for these extended ontologies. Each of them is identified, in the first column, by the number of disjunctive axioms that it contains. The second and third columns indicate the relative sizes of the computed $\mathcal{L}$-signature and the resulting $\mathcal{M}_{[\mathcal{O},\overline{\Sigma^{\mathcal{L}}}]}$. The last two columns give the classification times obtained using MORe and HermiT. In most cases classification times are improved by between one and two orders of magnitude.

We can observe, however, that axioms 15, 16, 18, and 19 have a significant effect on the size of $\Sigma^{\mathcal{L}}$ and, consequently, on the size of $\mathcal{M}_{[\mathcal{O},\overline{\Sigma^{\mathcal{L}}}]}$ and the total classification time. This is due to the classes involved in these particular axioms, which force our algorithm to keep very general classes outside $\Sigma^{\mathcal{L}}$; for example, in one of these cases, the class Liquid Substance is removed from the successive approximations to $\Sigma^{\mathcal{L}}$ at some point during its computation; by Proposition 2, all classes representing some kind of liquid substance—and therefore subsumed by the class Liquid Substance—must be left outside $\Sigma^{\mathcal{L}}$ too, which leads to a significantly smaller $\mathcal{L}$-signature. It is part of our future work plan to improve the heuristics we use in order to avoid, when possible, leaving out of $\Sigma^{\mathcal{L}}$ classes that are likely to be high up in the class subsumption hierarchy.

5 Conclusion and Future Work

In this paper, we have proposed a technique for classifying an OWL 2 ontology $\mathcal{O}$ by exploiting a reasoner for one of its profiles. Our technique allows us to show that the subsumers of many classes in $\mathcal{O}$ can be completely determined using only the fragment-specific reasoner. Our technique is general and flexible, it exhibits pay-as-you-go behaviour, and it is relatively easy to implement. Although the implementation in our reasoner MORe is still prototypical, our preliminary experiments show the potential of our approach in practice.

There are also many interesting possibilities for future work:

- Our heuristics for computing an $\mathcal{L}$-signature are rather basic, and there is plenty of room for improvement. For example, it might be possible to explore modular decomposition techniques to compute larger $\mathcal{L}$-signatures [24].
- $\bot$-modules provide very strong preservation guarantees (they preserve not just atomic subsumptions, but even *models*). It would be interesting to devise techniques for extracting modules that are more "permissive", in the sense that they only provide preservation guarantees for atomic subsumptions.
- Our technique could also be applied to a different notion of locality, as long as it satisfied a result analogous to Proposition 2.
- We could explore ontology rewriting techniques that complement module extraction. By rewriting $\mathcal{O}$ into an $\mathcal{L}$-ontology $\mathcal{O}'$ such that $\mathcal{O}' \models \mathcal{O}$, and classifying $\mathcal{O}'$, we can obtain an "upper bound" on the classification of $\mathcal{O}$.

Acknowledgements. This work was supported by the Royal Society, the EU FP7 project Optique and the EPSRC projects Score!, ExODA and LogMap.

References

1. Baader, F., Brandt, S., Lutz, C.: Pushing the $\mathcal{EL}$ envelope. In: IJCAI (2005)
2. Baader, F., Franconi, E., Hollunder, B., Nebel, B., Profitlich, H.: An empirical analysis of optimization techniques for terminological representation systems. Applied Intelligence 4(2), 109–132 (1994)

3. Baader, F., Lutz, C., Suntisrivaraporn, B.: CEL - a polynomial-time reasoner for life science ontologies. In: Proc. of IJCAR, pp. 287–291 (2006)
4. Cuenca Grau, B., Halaschek-Wiener, C., Kazakov, Y., Suntisrivaraporn, B.: Incremental classification of description logics ontologies. JAR 44(4), 337–369 (2010)
5. Cuenca Grau, B., Horrocks, I., Kazakov, Y., Sattler, U.: Modular reuse of ontologies: Theory and practice. JAIR 31, 273–318 (2008)
6. Cuenca Grau, B., Horrocks, I., Motik, B., Parsia, B., Patel-Schneider, P.F., Sattler, U.: OWL 2: The next step for OWL. J. Web Semantics (JWS) 6(4), 309–322 (2008)
7. Cuenca Grau, B., Horrocks, I., Kazakov, Y., Sattler, U.: Just the right amount: extracting modules from ontologies. In: WWW, pp. 717–726 (2007)
8. Glimm, B., Horrocks, I., Motik, B., Shearer, R., Stoilos, G.: A novel approach to ontology classification. J. of Web Semantics 10(1) (2011)
9. Haarslev, V., Möller, R.: High performance reasoning with very large knowledge bases: A practical case study. In: Proc. IJCAI, pp. 161–168 (2001)
10. Haarslev, V., Möller, R.: Racer system description. In: IJCAR, pp. 701–705 (2001)
11. Horrocks, I.: Implementation and optimisation techniques. In: The Description Logic Handbook: Theory, Implementation, and Applications, pp. 306–346 (2003)
12. Horrocks, I., Kutz, O., Sattler, U.: The even more irresistible $\mathcal{SROIQ}$. In: Proc. of KR, pp. 57–67 (2006)
13. Jiménez-Ruiz, E., Cuenca Grau, B.: LogMap: Logic-Based and Scalable Ontology Matching. 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. 273–288. Springer, Heidelberg (2011)
14. Jiménez-Ruiz, E., Cuenca Grau, B., Sattler, U., Schneider, T., Berlanga, R.: Safe and Economic Re-Use of Ontologies: A Logic-Based Methodology and Tool Support. In: Bechhofer, S., Hauswirth, M., Hoffmann, J., Koubarakis, M. (eds.) ESWC 2008. LNCS, vol. 5021, pp. 185–199. Springer, Heidelberg (2008)
15. Kazakov, Y.: $\mathcal{RIQ}$ and $\mathcal{SROIQ}$ are harder than $\mathcal{SHOIQ}$. In: Proc. of KR, pp. 274–284 (2008)
16. 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)
17. Lutz, C., Walther, D., Wolter, F.: Conservative extensions in expressive description logics. In: Proc. of IJCAI, pp. 453–458 (2007)
18. Lutz, C., Wolterinst, F.: Conservative Extensions in the Lightweight Description Logic $\mathcal{EL}$. In: Pfenning, F. (ed.) CADE 2007. LNCS (LNAI), vol. 4603, pp. 84–99. Springer, Heidelberg (2007)
19. Motik, B., Cuenca Grau, B., Horrocks, I., Wu, Z., Fokoue, A., Lutz, C.: OWL 2 web ontology language profiles. W3C Recommendation (2009)
20. OWL 2 Web Ontology Language Direct Semantics. W3C Recommendation (October 27, 2009), http://www.w3.org/TR/owl2-direct-semantics/
21. 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)
22. Tsarkov, D., Horrocks, I.: FaCT++ Description Logic Reasoner: System Description. In: Furbach, U., Shankar, N. (eds.) IJCAR 2006. LNCS (LNAI), vol. 4130, pp. 292–297. Springer, Heidelberg (2006)
23. Tsarkov, D., Horrocks, I., Patel-Schneider, P.: Optimizing terminological reasoning for expressive description logics. JAR 39(3), 277–316 (2007)
24. Vescovo, C.D., Parsia, B., Sattler, U., Schneider, T.: The modular structure of an ontology: Atomic decomposition. In: Proc. of IJCAI, pp. 2232–2237 (2011)

A Formal Semantics for Weighted Ontology Mappings

Manuel Atencia[1,2], Alexander Borgida[3], Jérôme Euzenat[1,2],
Chiara Ghidini[4], and Luciano Serafini[4]

[1] INRIA, France
{manuel.atencia,jerome.euzenat}@inria.fr
[2] University of Grenoble, France
[3] Rutgers University, United States
borgida@cs.rutgers.edu
[4] Fondazione Bruno Kessler, Italy
{ghidini,serafini}@fbk.eu

Abstract. Ontology mappings are often assigned a weight or confidence factor by matchers. Nonetheless, few semantic accounts have been given so far for such weights. This paper presents a formal semantics for weighted mappings between different ontologies. It is based on a classificational interpretation of mappings: if O_1 and O_2 are two ontologies used to classify a common set X, then mappings between O_1 and O_2 are interpreted to encode how elements of X classified in the concepts of O_1 are re-classified in the concepts of O_2, and weights are interpreted to measure how precise and complete re-classifications are. This semantics is justifiable by extensional practice of ontology matching. It is a conservative extension of a semantics of crisp mappings. The paper also includes properties that relate mapping entailment with description logic constructors.

1 Introduction

Ontology mappings are used to express semantic relations between components of two heterogeneous ontologies. They are key artifacts for the integration of knowledge encoded in distinct schemas. On the one hand, theoretical studies on ontology mappings give a formal background for *crisp mappings*, i.e. mappings that express set theoretical relations — subsumption ($\sqsubseteq$ and $\sqsupseteq$), equivalence ($\equiv$) and disjointness ($\perp$) — between the extensional meaning of the concepts and relations of two ontologies [3,16]. On the other hand, the majority of the state-of-the-art tools that automatically match ontologies generate *weighted mappings*, i.e., crisp mappings associated with a confidence value, typically a real number between 0 and 1 [5]. There is, however, no shared view on how these weights should be interpreted, neither in ontology matching nor in related fields such as database schema matching [6].

In this paper we fill this gap by providing a novel formal semantics for interpreting the confidence value associated with a mapping. This semantics is not based on standard probabilistic notions such as ones used to extend Description Logics (DL) and rule-based descriptions of uncertainty, but instead is based on a classificational interpretation of mappings which reflects a family of approaches used in ontology matching techniques (see Chapter 4.4. in [5], and [15] as a recent example): if two ontologies

O_1 and O_2 are used to classify a common set X of items, mappings between O_1 and O_2 encode how elements of X classified in the concepts of O_1 are re-classified in the concepts of O_2, and weights measure how precise and complete the re-classifications are. We fall back on precision, recall, and F-measures, as they are used in the context of classification tasks, for the formalisation of weighted subsumptions and equivalence, respectively.

The proposed semantics makes it possible to discover inconsistencies and detect implications over sets of weighted mappings. In other words, to understand when a mapping can be derived from others, and when sets of mappings are inconsistent. We introduce a notion of logical consequence between weighted mappings, and investigate entailment between weighted mappings w.r.t. description logic constructors. Moreover, we prove that the semantics for weighted mappings introduced in the paper is a conservative extension of the semantics of crisp mappings for a specific class of Distributed Description Logics (DDLs) [3].

The paper is organised as follows. Section 2 introduces a formal semantics for weighted mappings, including the notion of (weighted) mapping entailment. In Sections 3–5 we show the adequacy of the proposed semantics, including the proof that it is a conservative extension of the semantics of DDL mappings; properties of the mapping entailment w.r.t. description logic constructors; and variants of the proposed semantics. In Section 6, we summarise related work, and we finish with concluding remarks.

2 Classificational Semantics for Weighted Mappings

The semantics presented in this paper is designed for weighted mappings between pairs of ontologies expressed in (a fragment of) first order language with tarskian semantics. However, for the sake of presentation we focus on a particular fragment of FOL, namely the description logic $\mathcal{ALCO}$: the basic DL logic $\mathcal{ALC}$ extended with nominals. We choose $\mathcal{ALCO}$ to present this semantics, since it is the smallest logic that contains full propositional connectives, relations and constants. $\mathcal{ALCO}$ is a DL logic defined on an alphabet $\Sigma = \mathcal{CN} \uplus \mathcal{RN} \uplus \mathcal{ON}$, where $\mathcal{CN}$ is a set of concept symbols, $\mathcal{RN}$ is a set of role symbols, and $\mathcal{ON}$ is a set of individual symbols. Complex concept expressions (simply called concepts) in $\mathcal{ALCO}$ are defined by the following grammar:

$$C, D := \top | \bot | A | \neg C | C \sqcap D | \exists R.C | \{o\}$$

where $A \in \mathcal{CN}$, $R \in \mathcal{RN}$ and $o \in \mathcal{ON}$. As usual, $\forall R.C$ stands for $\neg\exists R.\neg C$, and $C \sqcup D$ for $\neg(\neg C \sqcap \neg D)$. A general inclusion axiom (GCI) is an expression of the form $C \sqsubseteq D$ where C and D are concepts.

An interpretation $\mathcal{I}$ is a pair $\mathcal{I} = \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$ where $\Delta^{\mathcal{I}}$ is a non-empty set, called the domain of $\mathcal{I}$ and $\cdot^{\mathcal{I}}$ is a function such that $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ for $A \in \mathcal{CN}$, $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ for $R \in \mathcal{RN}$, and $o^{\mathcal{I}} \in \Delta^{\mathcal{I}}$ for $o \in \mathcal{ON}$. The interpretation of complex concepts of $\mathcal{ALCO}$ is defined according to the following rules:

$$\begin{aligned} \top^{\mathcal{I}} &= \Delta^{\mathcal{I}} & (C \sqcap D)^{\mathcal{I}} &= C^{\mathcal{I}} \cap D^{\mathcal{I}} \\ \bot^{\mathcal{I}} &= \emptyset & \{a\}^{\mathcal{I}} &= \{a^{\mathcal{I}}\} \\ (\neg C)^{\mathcal{I}} &= \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}} & (\exists R.C)^{\mathcal{I}} &= \{x \in \Delta^{\mathcal{I}} | \exists y, (x, y) \in R^{\mathcal{I}} \;\&\; y \in C^{\mathcal{I}}\} \end{aligned}$$

An interpretation $\mathcal{I}$ satisfies a GCI $C \sqsubseteq D$, in symbols $\mathcal{I} \models C \sqsubseteq D$, if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$. An ontology O is a set of GCIs. An interpretation satisfies O, in symbols $\mathcal{I} \models O$, if $\mathcal{I} \models \varphi$ for all $\varphi \in O$. A CGI is entailed by an ontology O, in symbols $O \models C \sqsubseteq D$ if $\mathcal{I} \models C \sqsubseteq D$, for all interpretations $\mathcal{I}$ that satisfy O.

$\mathcal{ALCO}$ has the "finite model property", meaning that if an entailment holds in all finite interpretations (ones with finite domain), then it is a theorem, in the sense that it holds for all interpretations. Not all DLs have the finite model property, e.g., adding cardinality constraints and inverses to $\mathcal{ALCO}$ prevents it.

2.1 Weighted Mappings

We start from the definition of mapping presented in [3] that we recall below.

Definition 1. *Let $\{O_i\}_{i \in I}$ be a family of ontologies. A* mapping *from O_i to O_j is an expression of the form*

$$i : C \; r \; j : D$$

where C and D are concepts of O_i and O_j, respectively, and $r \in \{\sqsubseteq, \equiv, \sqsupseteq, \perp\}$.

The notion of *weighted mapping* generalises it by associating mappings to a closed subinterval of $[0, 1]$.

Definition 2. *Let $\{O_i\}_{i \in I}$ be a family of ontologies. A* weighted mapping *from O_i to O_j is an expression of the form*

$$i : C \; r_{[a,b]} \; j : D$$

where C and D are concepts of O_i and O_j, respectively, $r \in \{\sqsubseteq, \equiv, \sqsupseteq, \perp\}$ and a, b are real numbers in the unit interval $[0, 1]$.

Remark 1. Notice that if $a > b$ then $[a, b] = \emptyset$ which is a closed subinterval of $[0, 1]$.

2.2 Formal Semantics for Weighted Mappings

Our semantics for weighted mappings is based on the following intuition. Assume that the concepts of two ontologies O_1 and O_2 are used to classify a common set of elements X. Mappings from O_1 to O_2 encode how the elements of X classified in the concepts of O_1 are re-classified in the concepts of O_2, and the weights encode how precise and complete these re-classifications are. Let us pin down this intuition and see how it can be used to define a formal semantics for a weighted mapping $1 : C \; r_{[a,b]} \; 2 : D$.

Let $X = \{x_1, \ldots, x_n\}$ be a non-empty finite set of fresh constants not occurring in $L(O_1)$ or $L(O_2)$.[1] The set X is meant to represent the set of shared items classified by concepts of the ontologies O_1 and O_2. A classification of X in O_1 is specified by virtue of an interpretation $\mathcal{I}_1$ of O_1 extended with the elements of X as follows. Let C

[1] Be aware that X is not a concept name, and that $\{x_1, \ldots, x_n\}$ is not a concept. Instead X is a meta-notation used in this paper to refer to an arbitrary finite set.

be a concept of O_1 and x_k a fresh constant of X; we say that x_k is classified under C according to $\mathcal{I}_1$ if $x_k^{\mathcal{I}_1} \in C^{\mathcal{I}_1}$. The set

$$C_X^{\mathcal{I}_1} = \{x \in X \mid x^{\mathcal{I}_1} \in C^{\mathcal{I}_1}\}$$

then represents the subset of items of X classified under C according to $\mathcal{I}_1$. Note that $C_X^{\mathcal{I}_1}$ is a subset of X whereas $C^{\mathcal{I}_1}$ is a subset of the domain of the interpretation $\mathcal{I}_1$. In addition, $C_X^{\mathcal{I}_1}$ is always a finite set while $C^{\mathcal{I}_1}$ may be infinite.

Let $\mathcal{I}_1$ and $\mathcal{I}_2$ be interpretations of O_1 and O_2, respectively, and let C and D be the concepts of O_1 and O_2, occurring in $1{:}C\ r_{[a,b]}\ 2{:}D$. Since we do not want to make any commitment on the interpretation domains of the two ontologies, it may be the case that the sets $C^{\mathcal{I}_1}$ and $D^{\mathcal{I}_2}$ cannot be compared as they might be defined over independent interpretation domains. Yet the sets $C_X^{\mathcal{I}_1}$ and $D_X^{\mathcal{I}_2}$ can be compared as they are both subsets of X which represent the sets of items of X classified under C according to $\mathcal{I}_1$ and under D according to $\mathcal{I}_2$, respectively. We can therefore examine the different types of mappings $1{:}C\ r_{[a,b]}\ 2{:}D$ obtained by looking at the different $r \in \{\sqsubseteq, \equiv, \sqsupseteq, \perp\}$.

Intuitively, the mapping $1{:}C \sqsubseteq 2{:}D$ is used to express that any item in X which is classified under C according to $\mathcal{I}_1$ is (re-)classified under D according to $\mathcal{I}_2$. The weighted mapping $1{:}C \sqsubseteq_{[a,b]} 2{:}D$ is thus used to express the fact that the proportion of items of X classified under C according to $\mathcal{I}_1$ which are (re-)classified under D according to $\mathcal{I}_2$ lies in the interval $[a,b]$. Assuming that $|C_X^{\mathcal{I}_1}| \neq \emptyset$, we can rewrite this intuition in the formula

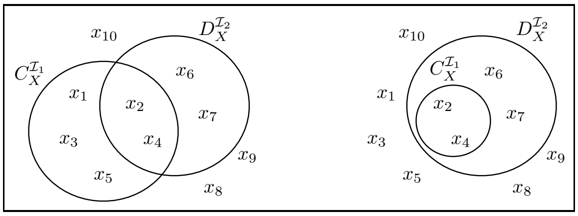
$$\frac{|C_X^{\mathcal{I}_1} \cap D_X^{\mathcal{I}_2}|}{|C_X^{\mathcal{I}_1}|} \in [a,b] \tag{1}$$

which can be seen as the recall of $D_X^{\mathcal{I}_2}$ w.r.t. $C_X^{\mathcal{I}_1}$. Indeed, given two sets A and B, the *recall* of B w.r.t. A is defined as

$$R(A,B) = \frac{|A \cap B|}{|A|}$$

unless $|A| = 0$, in which case $R(A,B) = 1$. Thus, the condition in (1) can be rephrased as $R(C_X^{\mathcal{I}_1}, D_X^{\mathcal{I}_2}) \in [a,b]$.

Example 1. Let $X = \{x_1, \ldots, x_{10}\}$, and $C_X^{\mathcal{I}_1}$ and $D_X^{\mathcal{I}_2}$ as in the two following diagrams:



It is immediate to see that on the left hand side diagram, $R(C_X^{\mathcal{I}_1}, D_X^{\mathcal{I}_2}) = \frac{2}{5} = 0.4$ while on the right hand side diagram, $R(C_X^{\mathcal{I}_1}, D_X^{\mathcal{I}_2}) = \frac{2}{2} = 1$.

The weighted mapping $1\!:\!C \sqsupseteq_{[a,b]} 2\!:\!D$, in turn, is used to express the fact that the fraction of items of X classified by D according to $\mathcal{I}_2$ which are (re-)classified under C according to $\mathcal{I}_1$ lies in the interval $[a, b]$. Under the assumption that $|D_X^{\mathcal{I}_2}| \neq \emptyset$, we can rewrite this as:

$$\frac{|C_X^{\mathcal{I}_1} \cap D_X^{\mathcal{I}_2}|}{|D_X^{\mathcal{I}_2}|} \in [a, b] \tag{2}$$

which can be seen as the precision of $D_X^{\mathcal{I}_2}$ w.r.t. $C_X^{\mathcal{I}_1}$. The *precision* of B w.r.t. A is in fact given by

$$P(A, B) = \frac{|A \cap B|}{|B|}$$

unless $|B| = 0$, in which case $P(A, B) = 1$. Thus, the condition in (2) can be rephrased as $P(C_X^{\mathcal{I}_1}, D_X^{\mathcal{I}_2}) \in [a, b]$.

Example 2. Let $X = \{x_1, \ldots, x_{10}\}$, and $C_X^{\mathcal{I}_1}$ and $D_X^{\mathcal{I}_2}$ as in Example 1. It is immediate to see that in both cases $P(C_X^{\mathcal{I}_1}, D_X^{\mathcal{I}_2}) = \frac{2}{4} = 0.5$.

By keeping the parallelism with classification systems, the natural way to interpret the weighted mapping $1\!:\!C \equiv_{[a,b]} 2\!:\!D$ is by means of the F-measure, which is the harmonic mean of precision and recall. The *F-measure* of A and B is defined as

$$F(A, B) = 2 \cdot \frac{P(A, B) \cdot R(A, B)}{P(A, B) + R(A, B)}$$

unless $P(A, B)$ and $R(A, B)$ are equal to 0, then $F(A, B) = 0$. It can be expressed as

$$F(A, B) = 2 \cdot \frac{|A \cap B|}{|A| + |B|}$$

unless $|A| = |B| = 0$, in which case $F(A, B) = 1$. In this way, the weighted mapping $1\!:\!C \equiv_{[a,b]} 2\!:\!D$ encodes that $F(C_X^{\mathcal{I}_1}, D_X^{\mathcal{I}_2}) \in [a, b]$.

We conclude the above explanation with the definition of the degree of satisfiability of a mapping. Then we introduce mapping satisfiability and mapping entailment.

Definition 3 (Degree of satisfiability of a mapping). *Let O_i and O_j be two ontologies and let X be a non-empty finite set of fresh individual constants. Let $\mathcal{I}_i$ and $\mathcal{I}_j$ be two interpretations of O_i and O_j, respectively, extended with the set X. The* degree of satisfiability of the mapping $i\!:\!C\ r\ j\!:\!D$ with respect to the pair $(\mathcal{I}_i, \mathcal{I}_j)$ and X *is denoted by* $\mathrm{ds}_X(\mathcal{I}_i, \mathcal{I}_j, C, r, D)$ *and defined as*

$$\begin{aligned}
\mathrm{ds}_X(\mathcal{I}_i, \mathcal{I}_j, C, \sqsubseteq, D) &= R(C_X^{\mathcal{I}_i}, D_X^{\mathcal{I}_j}) \\
\mathrm{ds}_X(\mathcal{I}_i, \mathcal{I}_j, C, \sqsupseteq, D) &= P(C_X^{\mathcal{I}_i}, D_X^{\mathcal{I}_j}) \\
\mathrm{ds}_X(\mathcal{I}_i, \mathcal{I}_j, C, \equiv, D) &= F(C_X^{\mathcal{I}_i}, D_X^{\mathcal{I}_j}) \\
\mathrm{ds}_X(\mathcal{I}_i, \mathcal{I}_j, C, \perp, D) &= 1 - F(C_X^{\mathcal{I}_i}, D_X^{\mathcal{I}_j})
\end{aligned}$$

The pair $(\mathcal{I}_i, \mathcal{I}_j)$ satisfies the weighted mapping $i\!:\!C\ r_{[a,b]}\ j\!:\!D$ modulo X, *denoted by* $(\mathcal{I}_i, \mathcal{I}_j) \models_X i\!:\!C\ r_{[a,b]}\ j\!:\!D$, *if and only if* $\mathrm{ds}_X(\mathcal{I}_i, \mathcal{I}_j, C, r, D) \in [a, b]$. *If M is a set of weighted mappings from O_i to O_j, the pair* $(\mathcal{I}_i, \mathcal{I}_j)$ satisfies M modulo X, *in symbols,* $(\mathcal{I}_i, \mathcal{I}_j) \models_X M$, *if* $(\mathcal{I}_i, \mathcal{I}_j) \models_X m$ *for every* $m \in M$.

Definition 4 (Mapping satisfiability). *Let O_i and O_j be two ontologies and let X be a non-empty finite set of fresh individual constants. Let M be a set of weighted mappings from O_i to O_j. The set M is* satisfiable modulo X *if there exist interpretations $\mathcal{I}_i$ and $\mathcal{I}_j$ of O_i and O_j, respectively, such that $(\mathcal{I}_i, \mathcal{I}_j) \models_X M^{ij}$. We say that the set M is* satisfiable *if there exists an $X \neq \emptyset$ such that M is satisfiable modulo X.*

Definition 5 (Mapping entailment). *Let O_i and O_j be two ontologies and let X be a non-empty finite set of fresh individual constants. Also, let M be a set of weighted mappings from O_i to O_j. The set M* entails $i\!:\!C\ r_{[a,b]}\ j\!:\!D$ modulo X, *denoted $M \models_X i\!:\!C\ r_{[a,b]}\ j\!:\!D$, if for every interpretations $\mathcal{I}_i$ and $\mathcal{I}_j$ of O_i and O_j, respectively, such that $(\mathcal{I}_i, \mathcal{I}_j)$ satisfies M modulo X, we have that $(\mathcal{I}_i, \mathcal{I}_j) \models_X i\!:\!C\ r_{[a,b]}\ j\!:\!D$. The set M* entails $i\!:\!C\ r_{[a,b]}\ j\!:\!D$, *in symbols, $M \models i\!:\!C\ r_{[a,b]}\ j\!:\!D$, if we have $M \models_X i : C\ r_{[a,b]}\ j\!:\!D$ for every $X \neq \emptyset$.*

Remark 2 (Inconsistent mappings). In the case of $a > b$, the mapping $i\!:\!C\ r_{[a,b]}\ j\!:\!D$ has no satisfying interpretations. That is, mappings defined over empty probability ranges are inconsistent mappings and we denote them by FALSE. This allows us to express unsatisfiability of a set of mappings in terms of mapping entailment: if M is a set of mappings between O_i to O_j, then $M \models FALSE$ is equivalent to stating that M is not satisfiable. Moreover, notice that $FALSE \models i\!:\!C\ r_{[a,b]}\ j\!:\!D$, i.e. every mapping is entailed by the inconsistent mapping.

3 Adequacy and Expressivity of the Semantics

In this section we prove the adequacy of the formal semantics for weighted mappings proposed in Section 2 by showing that (i) it is a conservative extension of a standard semantics for crisp mappings; (ii) it can be used to provide a formal interpretation of the results returned by automatic ontology matching algorithms; and (iii) it is general enough to provide a uniform formal interpretation of weighted mappings between both concepts and individuals.

3.1 Backward Compatibility with the Semantics of Crisp Mappings

In order to prove that the classificational semantics presented in this paper provides a suitable extension of the one for crisp mappings we show that it is a conservative extension of the DDL-based semantics for crisp mappings presented in [3] when the intervals $[a, b]$ are either $[0, 0]$ or $[1, 1]$. First of all, notice that when a and b are either 0 or 1 every mapping can be expressed in terms of $\sqsubseteq_{[1,1]}$. Indeed the following logical consequences hold:

- $i\!:\!C \sqsupseteq_{[1,1]} j\!:\!D$ is equivalent to $j\!:\!D \sqsubseteq_{[1,1]} i\!:\!C$
- $i\!:\!C \equiv_{[1,1]} j\!:\!D$ is equivalent to $i\!:\!C \sqsubseteq_{[1,1]} j\!:\!D$ and $j\!:\!D \sqsubseteq_{[1,1]} i\!:\!C$
- $i\!:\!C \perp_{[1,1]} j\!:\!D$ is equivalent to $i\!:\!C \sqsubseteq_{[1,1]} j\!:\!\neg D$
- $i\!:\!C \sqsubseteq_{[0,0]} j\!:\!D$ is equivalent to $i\!:\!C \sqsubseteq_{[1,1]} j\!:\!\neg D$
- $i\!:\!C \sqsupseteq_{[0,0]} j\!:\!D$ is equivalent to $i\!:\!C \sqsubseteq_{[1,1]} j\!:\!\neg D$
- $i\!:\!C \equiv_{[0,0]} j\!:\!D$ is equivalent to $i\!:\!C \sqsubseteq_{[1,1]} j\!:\!\neg D$

- $i\!:\!C \perp_{[0,0]} j\!:\!D$ is equivalent to $i\!:\!C \sqsubseteq_{[1,1]} j\!:\!D$ and $j\!:\!D \sqsubseteq_{[1,1]} i\!:\!C$
- $i\!:\!C\ r_{[0,1]}\ j\!:\!D$ is equivalent to $i\!:\!\bot \sqsubseteq_{[1,1]} j\!:\!\top$
- $i\!:\!C\ r_{[1,0]}\ j\!:\!D$ is equivalent to $i\!:\!\top \sqsubseteq_{[1,1]} j\!:\!\bot$

The above guarantees that any weighted mapping from O_i to O_j with weight in $\{[0,0],[0,1],[1,0],[1,1]\}$ can be rewritten as a $\sqsubseteq_{[1,1]}$-mapping between O_i and O_j (i.e. from O_i to O_j or from O_j to O_i). We say that a set of weighted mappings between O_i and O_j is in $\sqsubseteq_{[1,1]}$-normal form if they are of the form $x\!:\!C \sqsubseteq_{[1,1]} y\!:\!D$.

Lemma 1. *If M is a set of mappings between two ontologies O_i and O_j with weights in $\{[0,0],[0,1],[1,0],[1,1]\}$, then M can be rewritten in an equivalent set of mappings $M_{\sqsubseteq_{[1,1]}}$ in $\sqsubseteq_{[1,1]}$-normal form.*

Crisp mappings are defined in DDL via bridge rules, whose syntax and semantics are as follows: let $\mathcal{I}_i$ and $\mathcal{I}_j$ be interpretations of the ontologies O_i and O_j, resp. Let $\rho_{ij} \subseteq \Delta^{\mathcal{I}_i} \times \Delta^{\mathcal{I}_j}$ be a domain correspondence relation. In DDL we define four kinds of bridge rules, but, due to the rewriting described above, the only one that is interesting here is what the so-called into bridge rule:

$$(\mathcal{I}_i, \mathcal{I}_j, \rho_{ij}) \models i\!:\!C \xrightarrow{\sqsubseteq} j\!:\!D \quad \text{iff} \quad \rho_{ij}(C^{\mathcal{I}_i}) \subseteq D^{\mathcal{I}_j}$$

Lemma 2. *Let O_i and O_j be two ontologies.*
(1) Let $\mathcal{I}_i$ and $\mathcal{I}_j$ be two interpretations of O_i and O_j, resp., extended with a non-empty finite X. Then there exists a domain relation $\rho_{ij} \subseteq \Delta^{\mathcal{I}_i} \times \Delta^{\mathcal{I}_j}$ such that

$$(\mathcal{I}_i, \mathcal{I}_j) \models_X i\!:\!C \sqsubseteq_{[1,1]} j\!:\!D \quad \textit{iff} \quad (\mathcal{I}_i, \mathcal{I}_j, \rho_{ij}) \models i\!:\!C \xrightarrow{\sqsubseteq} j\!:\!D$$

(2) Let $\mathcal{I}_i$ and $\mathcal{I}_j$ be interpretations of O_i and O_j with finite domains $\Delta^{\mathcal{I}_i}$ *and* $\Delta^{\mathcal{I}_j}$, *and let $\rho_{ij} \subseteq \Delta^{\mathcal{I}_i} \times \Delta^{\mathcal{I}_j}$ be a* DDL *domain relation. Then there exists a non-empty finite set X of fresh individuals and interpretations $\mathcal{I}'_i$ and $\mathcal{I}'_j$ extending $\mathcal{I}_i$ and $\mathcal{I}_j$, resp., over X such that*

$$(\mathcal{I}_i, \mathcal{I}_j, \rho_{ij}) \models i\!:\!C \xrightarrow{\sqsubseteq} j\!:\!D \quad \textit{iff} \quad (\mathcal{I}'_i, \mathcal{I}'_j) \models_X i\!:\!C \sqsubseteq_{[1,1]} j\!:\!D$$

Proof (Outline). For (1) we define $\rho_{ij} = \{\langle x^{\mathcal{I}_i}, x^{\mathcal{I}_j}\rangle \mid x \in X\}$ and then show that $R(C_X^{\mathcal{I}_i}, D_X^{\mathcal{I}_j}) = 1$ iff $\rho_{ij}(C^{\mathcal{I}_i}) \subseteq D^{\mathcal{I}_j}$. In order to prove (2) we first choose a set of fresh constants $X = \{x_{(c,d)} \mid (c,d) \in \rho_{ij}\}$ (i.e one constant for each pair of the domain relation), and then extend $\mathcal{I}_i$ and $\mathcal{I}_j$ to $\mathcal{I}'_i$ and $\mathcal{I}'_j$ over X by defining $\mathcal{I}'_i(x_{(c,d)}) = c$ and $\mathcal{I}'_j(x_{(c,d)}) = d$. Then, $\rho_{ij}(C^{\mathcal{I}_i}) \subseteq D^{\mathcal{I}_j}$ iff $R(C_X^{\mathcal{I}'_i}, D_X^{\mathcal{I}'_j}) = 1$.

Theorem 1. *Consider ontologies O_i and O_j in ontology languages that have the finite-model property (recall that $\mathcal{ALCO}$ has this property). Let M be a set of mappings with weights in $\{[0,0],[0,1],[1,0],[1,1]\}$ and $M_{\sqsubseteq_{[1,1]}}$ its $\sqsubseteq_{[1,1]}$-normal form. Let $M_{\text{DDL}} = \{x\!:\!C \xrightarrow{\sqsubseteq} y\!:\!D \mid x\!:\!C \sqsubseteq_{[1,1]} y\!:\!D \in M_{\sqsubseteq_{[1,1]}}\}$ be the set of* corresponding DDL *bridge rules. Then, for arbitrary concepts C and D*

$$M \models i\!:\!C \sqsubseteq_{[1,1]} j\!:\!D \quad \textit{iff} \quad M_{\text{DDL}} \models_{inv} i\!:\!C \xrightarrow{\sqsubseteq} j\!:\!D$$

where $\models_{inv}$ denotes DDL *logical consequence restricted to models where $\rho_{ij} = \rho_{ji}^{-1}$.*

Proof (outline). Let us assume that $M_{\sqsubseteq_{[1,1]}} \not\models i\!:\!C \sqsubseteq_{[1,1]} j\!:\!D$. Then there exists a set $X \neq \emptyset$ and interpretations $\mathcal{I}_i$ and $\mathcal{I}_j$ of O_i and O_j, resp., extended with X such that $(\mathcal{I}_i, \mathcal{I}_j) \models_X M_{\sqsubseteq_{[1,1]}}$ but $(\mathcal{I}_i, \mathcal{I}_j) \not\models_X i\!:\!C \sqsubseteq_{[1,1]} j\!:\!D$. By Lemma 2, there is a domain relation ρ_{ij} for which $(\mathcal{I}_i, \mathcal{I}_j, \{\rho_{ij}, \rho_{ji}\}) \models M_{\text{DDL}}$ and $(\mathcal{I}_i, \mathcal{I}_j, \rho_{ij}) \not\models i\!:\!C \xrightarrow{r} j\!:\!D$. Therefore, $M_{\text{DDL}} \not\models i\!:\!C \xrightarrow{\sqsubseteq} j\!:\!D$.

Vice versa, assume $M_{\text{DDL}} \not\models i\!:\!C \xrightarrow{\sqsubseteq} j\!:\!D$. Thus, there is a DDL interpretation $(\mathcal{I}_i, \mathcal{I}_j, \{\rho_{ij}, \rho_{ji}\})$ with $\rho_{ij} = \rho_{ji}^{-1}$ such that $(\mathcal{I}_i, \mathcal{I}_j, \{\rho_{ij}, \rho_{ij}\}) \models M_{\text{DDL}}$ whereas $(\mathcal{I}_i, \mathcal{I}_j, \rho_{ij}) \not\models i\!:\!C \xrightarrow{\sqsubseteq} j\!:\!D$. One can extend the finite domain property of most DLs supporting qualified existential restrictions to those of DDLs using them via the "global DL" construction given in [3]. Therefore, we can suppose w.l.o.g that the domains of $\mathcal{I}_i$ and $\mathcal{I}_j$ are finite and, thus, ρ_{ij} and ρ_{ij} are also finite. By lemma 2, we have that there is an X such that $(\mathcal{I}_i, \mathcal{I}_j) \models_X M_{\sqsubseteq_{[1,1]}}$ but $(\mathcal{I}_i, \mathcal{I}_j) \not\models_X i\!:\!C \sqsubseteq_{[1,1]} j\!:\!D$. Notice that the fact that $\rho_{ij} = \rho_{ji}^{-1}$ guarantees that the Xs associated to ρ_{ij} and ρ_{ji} are the same.

3.2 Interpreting the Results of Ontology Matchers

The semantics presented in Section 2 expresses the weight of a mapping between two elements by means of an interval $[a, b]$, while ontology matching algorithms usually return a single confidence value c. Thus, we need to ask ourselves how we can represent this value c by means of the interval $[a, b]$. In answering this question we can opt for several alternatives: we can decide to represent c by means of the (pointwise) interval $[c, c]$, or the interval $[0, c]$, or the interval $[c, 1]$, or the interval $[c - \epsilon, c + \epsilon]$ centered in c. This flexibility of representation allows us to capture the different assumptions which are used by the different algorithms. Let us illustrate this by means of some examples. If we take a low confidence value c, some algorithms interpret it as an "I don't know" answer; others, instead, use it to represent the fact that "the two concepts are very different". These two usages of c can be captured in our formalism by two different encodings: in the first case c is formalized by the interval $[c, 1]$; in the second c corresponds to the interval $[0, c]$. A possible different representation is given when the result of an ontology matcher expresses an estimation of similarity with some degree of approximation. In this case the returned value c can be represented by means of the centered interval $[\max(0, c - \epsilon), \min(c + \epsilon, 1)]$, where ϵ is a value between 0 and 0.5 that depends on the level of accuracy of the matching algorithm: the more accurate the matcher is, the smaller the ϵ will be. As we can see from these few examples, the representation of c by means of the pointwise interval $[c, c]$ is only one among a set of different choices, and a very challenging one, since it says that the ontology matching algorithm returns the exact level of matching between two elements with a perfect level of accuracy (i.e. $\epsilon = 0$).

3.3 Uniform Semantics for Mapping between Concepts and Individuals

The semantics presented in Section 2 gives a uniform framework to interpret mappings that involve pairs of concepts, and pairs of individuals. If $C_X^{\mathcal{I}}$ is the set of elements of X that could be reclassified in C, $a_X^{\mathcal{I}}$ is the set of elements of X that could be the same as a. Contrary to $a^{\mathcal{I}}$ which is an element of the domain $\Delta_{\mathcal{I}}$, $a^{\mathcal{I}}_X$ is a subset of X. For this reason, we use the same notation for individual as for classes: $a^{\mathcal{I}}_X = \{a\}^{\mathcal{I}}_X$ and we use the notation $1:\{a\} \equiv_{[0.8,1]} 2:\{b\}$ to represent the fact a is "almost the same as" b. According to the formal semantics we have that

$$(\mathcal{I}_1, \mathcal{I}_2) \models_X 1:\{a\} \equiv_{[0.8,1]} 2:\{b\} \quad \text{iff} \quad 0.8 \leq 2 * \frac{|\{a\}^{\mathcal{I}_1}_X \cap \{b\}^{\mathcal{I}_2}_X|}{|\{a\}^{\mathcal{I}_1}_X| + |\{b\}^{\mathcal{I}_2}_X|} \leq 1$$

Therefore, the mapping $\{a\} \equiv_{[0.8,1]} \{b\}$ states that the harmonic mean of the fraction of items in X equivalent to a that are also equivalent to b, and the fraction of items in X equivalent to b which are also equivalent to a is between the numbers 0.8 and 1.

In order to better understand the intuition behind this formalization, consider the individuals Trento and TrentoTown belonging to two ontologies O_1 and O_2, resp. While there are interpretations $\mathcal{I}_1$ and $\mathcal{I}_2$ in which the two individuals coincide, that is, $\{\mathsf{Trento}\}^{\mathcal{I}_1}_X = \{\mathsf{TrentoTown}\}^{\mathcal{I}_2}_X$, there may be cases in which Trento is considered to be an area broader than TrentoTown but still largely overlapping with it. This can be formalized by considering two interpretations $\mathcal{I}_1$ and $\mathcal{I}_2$ where $\{\mathsf{Trento}\}^{\mathcal{I}_1}_X$ includes $\{\mathsf{TrentoTown}\}^{\mathcal{I}_2}_X$, for instance, $\{\mathsf{Trento}\}^{\mathcal{I}_1}_X = \{x_1, x_2, x_3, x_4\}$ and $\{\mathsf{TrentoTown}\}^{\mathcal{I}_2}_X = \{x_1, x_2, x_3\}$. In this case the mapping is weighted as

$$(\mathcal{I}_1, \mathcal{I}_2) \models_X 1:\{\mathsf{Trento}\} \equiv_{[0.85,0.86]} 2:\{\mathsf{TrentoTown}\}$$

where values 0.85 and 0.86 are obtained as under and over approximation of the fraction

$$2 * \frac{\left|\{\mathsf{Trento}\}^{\mathcal{I}_1}_X \cap \{\mathsf{TrentoTown}\}^{\mathcal{I}_2}_X\right|}{\left|\{\mathsf{Trento}\}^{\mathcal{I}_1}_X\right| + \left|\{\mathsf{TrentoTown}\}^{\mathcal{I}_2}_X\right|} = 2 * \frac{3}{7} = \frac{6}{7}$$

4 Properties of Mapping Entailment

In this section, we show how the proposed semantics can be used to compute additional mappings which are logical consequences of an initial set of mappings. The proofs are omitted for lack of space and can be found in [1].

Proposition 1 shows general properties of mapping entailment independently of the mapping relation considered.

Proposition 1. *The following hold:*

1. $\models i:A\ r_{[0,1]}\ j:G$
2. FALSE $\models i:A\ r_{[a,b]}\ j:G$
3. $i:A\ r_{[a,b]}\ j:G \models i:A\ r_{[c,d]}\ j:G$ *if* $[a,b] \subseteq [c,d]$.
4. $i:A\ r_{[a,b]}\ j:G, i:A\ r_{[c,d]}\ j:G \models i:A\ r_{[v,w]}\ j:G$, $v = \max(a,c)$, $w = \min(b,d)$.
5. $i:A\ r_{[a,b]}\ j:G \models j:G\ r^{-1}_{[a,b]}\ i:A$, *where* r^{-1} *is the inverse relation of* r.[2]

[2] Recall that $\equiv^{-1}$ is $\equiv$, $\sqsubseteq^{-1}$ is $\sqsupseteq$, $\sqsupseteq^{-1}$ is $\sqsubseteq$, and $\perp^{-1}$ is $\perp$.

Proposition 2 includes properties of mapping entailment which relate equivalence with subsumption and disjointness.

Proposition 2. *The following hold:*

1. $i\!:\!A \sqsubseteq_{[a,b]} j\!:\!G,\ i\!:\!A \sqsupseteq_{[c,d]} j\!:\!G \models i\!:\!A \equiv_{[v,w]} j\!:\!G$ *where*

$$v = \begin{cases} \frac{2ac}{a+c} & \text{if } a \neq 0 \text{ or } c \neq 0 \\ 0 & \text{if } a = c = 0 \end{cases} \quad \textit{and} \quad w = \begin{cases} \frac{2bd}{b+d} & \text{if } b \neq 0 \text{ or } d \neq 0 \\ 0 & \text{if } b = d = 0 \end{cases}$$

2. $i\!:\!A \equiv_{[a,b]} j\!:\!G \models i\!:\!A \sqsubseteq_{[v,1]} j\!:\!G$ *and*
   $i\!:\!A \equiv_{[a,b]} j\!:\!G \models i\!:\!A \sqsupseteq_{[v,1]} j\!:\!G$ *where* $v = \frac{a}{2-a}$
3. $i\!:\!A \equiv_{[a,b]} j\!:\!G,\ i\!:\!A \sqsubseteq_{[c,d]} j\!:\!G \models i\!:\!A \sqsupseteq_{[v,w]} j\!:\!G$ *and*
   $i\!:\!A \equiv_{[a,b]} j\!:\!G,\ i\!:\!A \sqsupseteq_{[c,d]} j\!:\!G \models i\!:\!A \sqsubseteq_{[v,w]} j\!:\!G$ *where*

$$v = \begin{cases} \frac{ac}{2d-a} & \text{if } a \neq 2d \\ 0 & \text{otherwise} \end{cases} \quad \textit{and} \quad w = \begin{cases} 0 & \text{if } b = 0 \\ \frac{bd}{2c-b} & \text{if } b \neq 0 \text{ and } b < \frac{2c}{1+d} \\ 1 & \text{otherwise} \end{cases}$$

4. $i\!:\!A \equiv_{[a,b]} j\!:\!G \models i\!:\!A \perp_{[v,w]} j\!:\!G$ *and*
   $i\!:\!A \perp_{[a,b]} j\!:\!G \models i\!:\!A \equiv_{[v,w]} j\!:\!G$ *where* $v = 1 - b$ *and* $w = 1 - a$

Proposition 3 introduces properties of mapping entailment with respect to $\mathcal{ALCO}$ constructors. No property about existential restriction is included, since, as mentioned before, mappings between roles are not considered in this paper.

Proposition 3. *The following hold:*

1. $i\!:\!A \sqsubseteq_{[a,b]} j\!:\!G \models i\!:\!A \sqsubseteq_{[v,w]} j\!:\!\neg G$ *where* $v = 1 - b$ *and* $w = 1 - a$
2. $\left.\begin{array}{l} i\!:\!A \sqsubseteq_{[a,b]} j\!:\!G \\ i\!:\!A \sqsupseteq_{[c,d]} j\!:\!\top \\ i\!:\!\top \sqsubseteq_{[e,f]} j\!:\!G \end{array}\right\} \models i\!:\!\neg A \sqsubseteq_{[v,w]} j\!:\!G$ *where*

$$v = \begin{cases} \max\left(0, \frac{e-bd}{1-c}\right) & \text{if } c \neq 1 \\ 1 & \text{if } c = 1 \end{cases} \quad \textit{and} \quad w = \begin{cases} \min\left(\frac{f-ac}{1-d}, 1\right) & \text{if } d \neq 1 \\ 1 & \text{if } d = 1 \end{cases}$$

3. $\left.\begin{array}{l} i\!:\!A \sqsubseteq_{[a,b]} j\!:\!G \\ i\!:\!A \sqsubseteq_{[c,d]} j\!:\!H \end{array}\right\} \models i\!:\!A \sqsubseteq_{[v,w]} j\!:\!G \sqcap H$ *where* $\begin{cases} v = \max(0, a+c-1), \ \textit{and} \\ w = \min(b,d) \end{cases}$
4. $\left.\begin{array}{l} i\!:\!A \sqsubseteq_{[a,b]} j\!:\!G \\ i\!:\!A \sqcap B \sqsupseteq_{[c,d]} j\!:\!G \end{array}\right\} \models i\!:\!A \sqcap B \sqsubseteq_{[v,1]} j\!:\!G$ *where* $v = ac$
5. $i\!:\!A \sqsubseteq_{[a,b]} j\!:\!G \sqcap H \models i\!:\!A \sqsubseteq_{[a,1]} j\!:\!G$
6. $\left.\begin{array}{l} i\!:\!A \sqcap B \sqsubseteq_{[a,b]} j\!:\!G \\ i\!:\!A \sqcap B \sqsupseteq_{[c,d]} j\!:\!G \end{array}\right\} \models i\!:\!A \sqsubseteq_{[0,w]} j\!:\!G$ *where* $w = \frac{b}{c}$
7. $i\!:\!A \ \sqsubseteq_{[a,b]} \ j\!:\!\{g\} \models i\!:\!A \ \sqsubseteq_{[0,b]} \ j\!:\!\{g\} \sqcap G$
8. $i\!:\!\{a\} \ \sqsubseteq_{[a,b]} \ j\!:\!G \models i\!:\!\{a\} \sqcap A \ \sqsubseteq_{[a,1]} \ j\!:\!G$

Notice that proposition 3 does not contain any rule that allow to infer some weight interval $[v, w]$ of the mapping $1 : A \sqcap B \sqsubseteq_{[v,w]} 2 : C$ starting from the weight intervals $[a, a']$ and $[b, b']$ of the mappings $i : A \sqsubseteq_{[a,a']} 2 : C$ and $i : B \sqsubseteq_{[b,b']} 2 : C$. This contrasts to what happens for crisp mappings, where $1 : A \sqcap B \sqsubseteq_{[1,1]} 2 : C$ is a logical consequence of the two mappings $i : A \sqsubseteq_{[1,1]} 2 : C$ and $i : B \sqsubseteq_{[1,1]} 2 : C$. In general there is an independence between the weights associated to the mapping of two concepts and the weight associated to the mapping of their conjunction. Example 3 provides an evidence of this independence by showing that mappings of the form $i : A \sqsubseteq_{[x,x]} 2 : C$ and $i : B \sqsubseteq_{[y,y]} 2 : C$ with high (resp. low) values of x and y are consistent with mappings of the form $1 : A \sqcap B \sqsubseteq_{[z,z]} 2 : C$ with a low (resp. high) value for z.

Example 3. Suppose that O_1 contains the concepts Professor and Professional, while O_2 includes the concepts HasFreeTime and EarnsALot. The following two interpretations $\mathcal{I}_1$ and $\mathcal{I}_2$

$$\mathsf{Professor}_X^{\mathcal{I}_1} = \{x_1, \ldots, x_{100}\} \qquad \mathsf{HasFreeTime}_X^{\mathcal{I}_2} = \{x_1, \ldots, x_{90}, x_{101}, \ldots, x_{190}\}$$
$$\mathsf{Professional}_X^{\mathcal{I}_1} = \{x_{91}, \ldots, x_{190}\} \qquad \mathsf{EarnsALot}_X^{\mathcal{I}_2} = \{x_{91}, \ldots, x_{100}\}$$

satisfies the following mappings:

$$(\mathcal{I}_1, \mathcal{I}_2) \models_X 1{:}\mathsf{Professor} \sqsubseteq_{[0.9,0.9]} 2{:}\mathsf{HasFreeTime} \tag{3}$$
$$(\mathcal{I}_1, \mathcal{I}_2) \models_X 1{:}\mathsf{Professional} \sqsubseteq_{[0.9,0.9]} 2{:}\mathsf{HasFreeTime} \tag{4}$$
$$(\mathcal{I}_1, \mathcal{I}_2) \models_X 1{:}\mathsf{Professor} \sqcap \mathsf{Professional} \sqsubseteq_{[0.1,0.1]} 2{:}\mathsf{HasFreeTime} \tag{5}$$
$$(\mathcal{I}_1, \mathcal{I}_2) \models_X 1{:}\mathsf{Professor} \sqsubseteq_{[0.1,0.1]} 2{:}\mathsf{EarnsALot} \tag{6}$$
$$(\mathcal{I}_1, \mathcal{I}_2) \models_X 1{:}\mathsf{Professional} \sqsubseteq_{[0.1,0.1]} 2{:}\mathsf{EarnsALot} \tag{7}$$
$$(\mathcal{I}_1, \mathcal{I}_2) \models_X 1{:}\mathsf{Professor} \sqcap \mathsf{Professional} \sqsubseteq_{[0.9,0.9]} 2{:}\mathsf{EarnsALot} \tag{8}$$

Notice that $\mathcal{I}_1$ and $\mathcal{I}_2$ satisfy the low weight mapping (5), on the conjunction of two concepts, as well as the high weight mappings (3) and (4) defined on the two conjunct concepts. Conversely, the two interpretations satisfy the high weight mapping (8) and the low weight mapping (6) and (7).

Proposition 4 shows mapping entailments in the presence of local knowledge. Local entailment in ontology O_i is denoted by $\models_i$. As in the case of weighted mappings, if we write $\models_i C \; r \; D$, we assume that C and D belong to $L(O_i)$.

Proposition 4. *The following hold:*

1. *If* $\models_i A \equiv B$ *and* $\models_j G \equiv H$ *then* $i{:}A \; r_{[v,w]} \; j{:}G \models i{:}B \; r_{[v,w]} \; j{:}H$
2. *If* $\models_i A \sqsubseteq B$ *then* $i{:}A \sqsupseteq_{[v,w]} j{:}G \models i{:}B \sqsupseteq_{[v,1]} j{:}G$
3. *If* $\models_i A \sqsupseteq B$ *then* $i{:}A \sqsupseteq_{[v,w]} j{:}G \models i{:}B \sqsupseteq_{[0,w]} j{:}G$
4. *If* $\models_i A \sqcap B \sqsubseteq \bot$ *then* $i{:}A \sqsupseteq_{[v,w]} j{:}G \models i{:}B \sqsupseteq_{[0,1-v]} j{:}G$

5 Variations on a Theme

There are a number of places above where we have made certain choices that could have been done differently. We examine some of the alternatives in this section.

5.1 Interpreting Weighted Equivalence Mappings

The choice of interpreting the weighted equivalence mapping by means of the F-measure is based on the fact that this is the typical measure used to evaluate the global quality of a classifier. However, any function $f : [0,1]^2 \to [0,1]$ that satisfies the following properties can be chosen to combine specific precision and recall values:

1. $f(0,0) = f(1,0) = f(0,1) = 0$, and $f(1,1) = 1$. The motivation for this is so that in the case of crisp mappings, the weight of an equivalence mapping be the logical "and" of the weights of the subsumption mappings.
2. $f(\cdot,\cdot)$ is monotonic in each variable. In other words, if the weight of a subsumption mapping increases then the value of the equivalence mapping must also increase.

In this paper, $f(\cdot,\cdot)$ is the F-measure, but parallel studies can be conducted where, for instance, $f(x,y) = \max(x,y)$, $f(x,y) = x \cdot y$ or $f(x,y) = \frac{x+y}{2}$.

A different approach to interpret the weighted equivalence mapping, starts from the usual definition of equivalence in DLs in terms of subsumption: ($A \equiv B$) iff ($A \sqsubseteq B$) and ($A \sqsupseteq B$). When dealing with single numbers for precision and recall, it is usually imposible to combine them into a single value by simple conjunction; hence the use of F as above. However, when using *ranges* of scores $[a,b]$ for subsumption relations, one can define $A \equiv^{\vee}_{[a,b]} B$ as the conjunction of $A \sqsubseteq_{[a,b]} B$ and $A \sqsupseteq_{[a,b]} B$. This leads to a natural rule: if $A \sqsubseteq_{[a_1,b_1]} B$ and $A \sqsupseteq_{[a_2,b_2]} B$ then $A \equiv^{\vee}_{[v,w]} B$ with $v = \min(a_1,a_2)$ and $w = \max(b_1,b_2)$.

The current version of our semantics is characterised by the fact that it allows a single individual in ontology O_1 to be matched with certainty to *sets* of individuals in O_2. That is, we can express $\{1{:}\mathit{book}1\} \equiv_{[1,1]} \{2{:}\mathit{book}1\_\mathit{copy}1, 2{:}\mathit{book}1\_\mathit{copy}2\}$. This is certainly useful if we want to express the fact that a book (e.g. *book*1) is equivalent to all the copies of that book. On the other hand, as pointed out by Cuenca Grau et al [7] the DDL which corresponds to our semantics, constructed according to Theorem 1 has certain conclusions that may sometimes be undesirable. The example in [7] starts from an ontology O_1 which states that the concepts Flying and NoNFlying are disjoint and that all birds can fly, and an ontology O_2 which defines the concept Penguin, and it connects the two ontologies with DDL bridge rules which map Bird onto Penguin and NoNFlying onto Penguin as described below:

$$
\begin{array}{ll}
1{:}\mathsf{NoNFlying} \sqcap \mathsf{Flying} \sqsubseteq \bot, & 1{:}\mathsf{Bird} \xrightarrow{\sqsupseteq} 2{:}\mathsf{Penguin}, \\
1{:}\mathsf{Bird} \sqsubseteq \mathsf{Flying}, & 1{:}\mathsf{NoNFlying} \xrightarrow{\sqsupseteq} 2{:}\mathsf{Penguin} \\
2{:}\mathsf{Penguin} \sqsubseteq \top, &
\end{array}
$$

The point made in [7] is that if we describe this example in one ontology, by rewriting $\xrightarrow{\sqsubseteq}$ and $\xrightarrow{\sqsupseteq}$ mappings by means of $\sqsubseteq$ and $\sqsupseteq$ statements, we obtain an unsatisfiable Penguin concept, as penguins cannot be (flying) birds and non flying creatures at the same time. In DDL the two ontologies O_1, O_2 and the above mappings are satisfiable. Intuitively this happens because a penguin x in ontology O_2 can correspond, via the DDL domain relation, to two distinct objects in O_1: one flying, and the other not.

This effect can be avoided by restricting the domain relation in DDL to be 1-to-1, as in P-DL [2]. A similar restriction can also be imposed to the semantics of weighted mappings given in Section 2: one simply requires that the interpretations $\mathcal{I}_1$ and $\mathcal{I}_2$ be 1-to-1 on the set X.[3] It can easily be verified that the corresponding proofs of Lemma 2 and of Theorem 1 go through. Note that in the case of 1-to-1 relations, all satisfiable mappings between individuals can be reduced to mappings in the intervals $[0, 0]$, $[1, 1]$ or $[0, 1]$. That because the value of precision, recall, and F-measure in this case is either 0 or 1.

5.2 A General Framework for Probabilistic Mappings

A natural question to ask is why not simply union the two ontologies O_1 and O_2 with the precision and recall statements expressed as probabilistic subsumptions between their terms, into one "global" probabilistic DL ontology (PDLO), and then reason with it. Our answer is that intuitively one wants to keep their domains of interpretation disjoint. The set X, whose identifiers did not appear in either O_1 nor O_2, but were independently interpreted into the domains of O_1 and O_2 respectively, played a crucial role in this.

We show here how one could create a global PDLO which respects this intuition, and from which one could draw a variety of conclusions depending on which specific probabilistic DL was chosen. (See Lukasiewicz and Straccia's review [13] for a variety of proposals.) The idea of such a translation is inspired by our earlier work on DDL [3], where we also constructed a single global DL, though the details of our construction here are different.

Let us suppose that O_1 and O_2 are two ontologies and let $P = \{1\!:\!A_k \sqsubseteq_{[a,b]} 2\!:\!G_k\}$ be a set of weighted inclusion mapping statements between O_1 and O_2.

Suppose that $\mathcal{DL}\_\mathcal{P}$ is some probabilistic description logic, where one can make inequality assertions on the probability of subsumptions of the form (E $\sqsubseteq$ F) *with probability* $\lesseqgtr p$.

Then consider the following translation from O_1, O_2 and P into a $\mathcal{DL}\_\mathcal{P}$ TBox T_{12}:

- The atomic concept symbols of T_{12} consist of $\{1\!:\!A \mid A \in CN_1\} \cup \{2\!:\!G \mid G \in CN_2\} \cup \{1\!:\!\mathsf{ANYTHING}, 2\!:\!\mathsf{ANYTHING}, X\}$.
- The atomic role symbols of T_{12} consist of $\{1 : R \mid R \in RN_1\} \cup \{2 : S \mid S \in RN_2\} \cup \{\rho_1, \rho_2\}$.
- The axioms of T_{12} include
  - relabelled local axioms of O_1 and O_2, obtained by prefixing all identifiers and subconcepts with 1 and 2 respectively.
  - transformations of all concepts and axioms, as described in [3], so that complementation in O_i is with respect to $i\!:\!\mathsf{ANYTHING}$ only (e.g., $\neg 1\!:\!A$ is replaced by $1\!:\!\mathsf{ANYTHING} \sqcap \neg 1\!:\!A$).
  - $(1 : \mathsf{ANYTHING} \sqcap 2 : \mathsf{ANYTHING} \sqsubseteq \bot)$, expressing the disjointness of the domains of the two ontologies being mapped.
  - axioms restricting ρ_i to have domain X and range $i\!:\!\mathsf{ANYTHING}$ for $i = 1, 2$. The idea is that ρ_i will play the role of the interpretation $\mathcal{I}_i$ applied to X.

[3] This corresponds to requiring the values in X to obey the unique name assumption.

- for every statement $A \sqsubseteq_{[a,b]} G$ add $(\exists\rho_1.1 : A \sqsubseteq \exists\rho_2.2 : G)$ *with probability* $\geq a$ and $(\exists\rho_1.1 : A \sqsubseteq \exists\rho_2.2 : G)$ *with probability* $\leq b$
  Essentially, these axioms establish the probabilistic subset relationships between the elements of X mapped by ρ_i
- converse axioms for every statement $A \sqsupseteq_{[a,b]} G$, adding $(\exists\rho_2.2 : G \sqsubseteq \exists\rho_1.1 : A)$ *with probability* $\geq a$ and $(\exists\rho_2.2 : G \sqsubseteq \exists\rho_1.1 : A)$ *with probability* $\leq b$
- If $\mathcal{DL\_P}$ requires all axioms to be stated with a probability, add probability 1 to the axioms built before the last 2 steps.
- In case one wants to simulate the approach presented in this paper, one would also like to enforce that X is a finite set. If the $\mathcal{DL\_P}$ has the finite model property, then we are safe since we can restrict ourselves to reasoning in finite models. Otherwise, if X could be made finite using axioms then so could the domain concept, $\top$, and it is known that reasoning in only finite models leads to very different and complex deductions [4].

We believe the above framework (obviously adjusted for differences like the use of Bayes nets for probabilistic statements) will allow us to explore in the future the differences/similarities and benefits/tradeoffs between various ways of defining weighted mappings based on the numerous proposals for probabilistic DLs in the literature, as well as the one introduced in this paper. The first candidate for such an investigation will be Heinsohn's pioneering proposal for probabilistic DL, $\mathcal{ALCP}$ [9]. Heinsohn models uncertain subsumption between concepts as conditional probabilities — what he calls p-conditionings — and also considers intervals instead of single values.

6 Related Work

In the last years there has been a growing interest in the management of uncertainty and vagueness in DLs for the semantic web [13]. Two main lines of research have been followed: probabilistic generalisations of DLs to deal with uncertainty, and also fuzzy extensions of DLs to handle vagueness. Since our approach falls into the first category we only report, in the presence of comprehensive surveys such that [13], some of the previous attempts to combine DLs with probability, paying special attention to those related to ontology mapping.

Heinsohn was one of the first to provide a probabilistic extension of description logics [9]. The language $\mathcal{ACLP}$ builds over $\mathcal{ACL}$ and adds probabilistic subsumptions between concepts. These are formalised in terms of the so-called p-conditionings which encode that the conditional probability of a concept given another concept lies in a concrete real interval. Heinsohn assumes interpretation domains to be finite, and defines conditional probabilities in terms of set cardinalities.

Koller et al. presented P-CLASSIC [10] which is a probabilistic generalisation of a version of the description logic CLASSIC. P-CLASSIC follows Heinsohn's approach and aims at answering probabilistic subsumption queries, but its semantics is based on a reduction to Bayesian networks.

More recently, Lukasiewicz proposed probabilistic extensions of $\mathcal{SHOIN}(\mathbf{D})$ and $\mathcal{SHIF}(\mathbf{D})$ [11]. Uncertain knowledge is realised by way of conditional constraints which, similarly to p-conditionings, encode interval restrictions for conditional probabilities over concepts. The semantics is based on the notion of lexicographic entailment in probabilistic default reasoning. Conditional constraints can be applied to individuals too. Thus, it is possible to represent, besides statistical knowledge about concepts, degrees of belief about individuals.

Although these approaches do not tackle the formalisation of ontology mappings directly, they certainly could be used for this purpose. Given two ontologies O_1 and O_2 one could express probabilistic subsumptions between their concepts within one "global" probabilistic ontology (in one of the formalisms described above), and then reason with it. The main motivation of this work is to provide a formalism that keeps the domains of the ontology interpretations disjoint, as it is done in [3,16] for crisp mappings. The classificational interpretation of mappings is itself a novel approach and justifies the use of F-measure to model equivalence mappings which could not be addressed with most of the existing probabilistic description logics.

From a very different perspective, Lukasiewicz et al. presents in [12] a language for representing and reasoning with uncertain ontology mappings. This approach is based on the tight integration of disjunctive logic programs under the answer set semantics, the description logics $\mathcal{SHOIN}(\mathbf{D})$ and $\mathcal{SHIF}(\mathbf{D})$, and Bayesian probabilities.

The work by Lutz and Schröder [14] introduces a family of probabilistic DLs the members of which relate to the probabilistic FOL of [8] in the same way as classical DLs relate to FOL. This family, denoted by Prob-DLs, introduces a set of probabilistic constructors $P_{\sim p}$ where $\sim \in \{<, \leq, =, \geq, >\}$ and $p \in [0, 1]$ to be applied to concepts and in some cases also to roles. If C is a concept, then $P_{\sim p}C$ is a concept which denotes objects that are an instance of C with probability $\sim p$. The semantics of Prob-DLs is based on probabilistic interpretations, which extend a classical DL interpretation with a probability distribution over a set of possible worlds. Concept subsumption in Prob-DLs refers to classical DL concept subsumption, although in every world. It is claimed that Prob-DLs are well-suited to capture aspects of uncertainty that are present in almost all biomedical ontologies. We believe, though, that they do not capture the intended semantics of ontology mappings.

7 Conclusions

Distributed ontology mappings are highly uncertain. We investigated the possibility to provide a reclassification semantics for weighted mappings extending DDL semantics. Reclassification semantics is based on the probability that individuals classified under a particular concept in one ontology would be classified in another concept in another ontology. Such a probabilistic view on weights should match the practice of matchers based on concept extensions or relation to a wider context, e.g., annotated resources.

We showed that such a semantics was preserving the classical DDL semantics in the sense that if crisp DDL mappings are encoded as weighted mappings with $[1, 1]$ weights, the consequences correspond. In fact, the reclassification semantics may be used as an alternative semantics to classical DDL semantics.

Then, weighted mapping entailment was defined from this semantics. Inferred mappings predict the probability of reclassification from weighted mappings. This allows one to infer mappings across different mapping relations and term constructions. It also relates mapping inference to (crisp) ontological reasoning. As usual with probabilistic approaches, precision weakens with inference. We also discussed variation of the proposed framework.

There are several avenues for this work. The most direct one is to apply it to mapping (or ontology) debugging by ranking given and inferred mappings according to their weight intervals and help to detect those near mappings which would not appear as crisp mappings but are mappings of high weight.

Acknowledgements. Manuel Atencia is partially supported under Qualinca project (ANR-2012-CORD-012) sponsored by the French National Research Agency.

References

1. Atencia, M., Borgida, A., Euzenat, J., Ghidini, C., Serafini, L.: A formal semantics for weighted ontology mappings. Tech. Rep. 81401, Fondazione Bruno Kessler - IRST (2012)
2. Bao, J., Voutsadakis, G., Slutzki, G., Honavar, V.: Package-Based Description Logics. In: Stuckenschmidt, H., Parent, C., Spaccapietra, S. (eds.) Modular Ontologies. LNCS, vol. 5445, pp. 349–371. Springer, Heidelberg (2009)
3. Borgida, A., Serafini, L.: Distributed Description Logics: Assimilating Information from Peer Sources. In: Spaccapietra, S., March, S., Aberer, K. (eds.) Journal on Data Semantics I. LNCS, vol. 2800, pp. 153–184. Springer, Heidelberg (2003)
4. Calvanese, D.: Unrestricted and finite model reasoning in class-based representation formalisms. Number viii-96-2 of collana delle tesi del dottorato di ricerca in informatica, Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza" (1996)
5. Euzenat, J., Shvaiko, P.: Ontology matching. Springer (2007)
6. Gal, A.: Uncertain Schema Matching. Synthesis Lectures on Data Management. Morgan & Claypool (2011)
7. Grau, B.C., Parsia, B., Sirin, E.: Combining owl ontologies using epsilon-connections. Journal of Web Semantics 4(1), 40–59 (2006)
8. Halpern, J.Y.: Reasoning about uncertainty. MIT Press (2003)
9. Heinsohn, J.: Probabilistic description logics. In: UAI 1994: Proceedings of the 10th Annual Conference on Uncertainty in Artificial Intelligence, pp. 311–318. Morgan Kaufmann (1994)
10. Koller, D., Levy, A.Y., Pfeffer, A.: P-classic: A tractable probablistic description logic. In: Proceedings of the 14th National Conference on Artificial Intelligence (AAAI 1997), pp. 390–397. AAAI Press (1997)
11. Lukasiewicz, T.: Expressive probabilistic description logics. Artificial Intelligence 172(6-7), 852–883 (2008)
12. Lukasiewicz, T., Predoiu, L., Stuckenschmidt, H.: Tightly integrated probabilistic description logic programs for representing ontology mappings. Annals of Mathematics and Artificial Intelligence 63(3-4), 385–425 (2011)
13. Lukasiewicz, T., Straccia, U.: Managing uncertainty and vagueness in description logics for the Semantic Web. Journal of Web Semantics 6(4), 291–308 (2008)

14. Lutz, C., Schröder, L.: Probabilistic description logics for subjective uncertainty. In: Proceedings of the 12th International Conference on the Principles of Knowledge Representation and Reasoning (KR 2010). AAAI Press (2010)
15. Tournaire, R., Petit, J.-M., Rousset, M.-C., Termier, A.: Discovery of Probabilistic Mappings between Taxonomies: Principles and Experiments. In: Spaccapietra, S. (ed.) Journal on Data Semantics XV. LNCS, vol. 6720, pp. 66–101. Springer, Heidelberg (2011)
16. Zimmermann, A., Euzenat, J.: Three Semantics for Distributed Systems and Their Relations with Alignment 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. 16–29. Springer, Heidelberg (2006)

Personalised Graph-Based Selection of Web APIs

Milan Dojchinovski[1], Jaroslav Kuchar[1], Tomas Vitvar[1], and Maciej Zaremba[2]

[1] Web Engineering Group
Faculty of Information Technology
Czech Technical University in Prague
firstname.lastname@fit.cvut.cz

[2] Digital Enterprise Research Institute
National University of Ireland, Galway
maciej.zaremba@deri.org

Abstract. Modelling and understanding various contexts of users is important to enable personalised selection of Web APIs in directories such as Programmable Web. Currently, relationships between users and Web APIs are not clearly understood and utilized by existing selection approaches. In this paper, we present a semantic model of a Web API directory graph that captures relationships such as Web APIs, mashups, developers, and categories. We describe a novel configurable graph-based method for selection of Web APIs with personalised and temporal aspects. The method allows users to get more control over their preferences and recommended Web APIs while they can exploit information about their social links and preferences. We evaluate the method on a real-world dataset from ProgrammableWeb.com, and show that it provides more contextualised results than currently available popularity-based rankings.

Keywords: Web APIs, Web services, personalisation, ranking, service selection, social network.

1 Introduction

The rapid growth of Web APIs and a popularity of service-centric architectures promote a Web API as a core feature of any Web application. According to ProgrammableWeb[1], a leading service and mashup directory, the number of Web APIs has steadily increased since 2008. While it took eight years to reach 1,000 APIs in 2008, and two years to reach 3,000 in 2010, it took only 10 months to reach 5,000 by the end of 2011 [16]. In spite of this increase, several problems are starting to arise. Old and new not yet popular Web APIs usually suffer from the preferential attachment problem [14], developers can only run a keyword-based search in a service directory or they run a Google search to find Web pages that reference or describe Web APIs. Although there exist a number of sophisticated

[1] http://www.programmableweb.com

mechanisms for service discovery, selection and ranking, there is still a lack of methods that would in particular take into account a wider Web APIs' and developers' contexts including developers' profiles, information who developed Web APIs or used them in a mashup, Web APIs' or mashups' categories as well as the time when an API or a mashup was developed or published. With the popularity of Web APIs and directory services like ProgrammableWeb, it is now possible to utilize all such information in more sophisticated service selection methods.

In this paper we develop a novel Web API selection method that provides personalized recommendations. As an underlying dataset we create so called *Linked Web APIs*, an RDF representation of the data from the ProgrammableWeb directory, that utilizes several well-known RDF vocabularies. The method has the following characteristics: 1) *social and linked*–it exploits relationships among Web APIs, mashups, categories, and social relationships among developers such as who knows who in the ProgrammableWeb directory, 2) *personalized*–it takes into account user's preferences such as developers the user knows and preferences that define importances of predicates, and 3) *temporal*–it takes into account a time when Web APIs and mashups appeared in the graph for the first time.

We develop a method called the *Maximum Activation* and show how it can be used for the Web API selection. The method calculates a maximum activation from initial nodes of the graph (defined by a user profile), to each node from a set where a node in the set represents a Web API candidate. We adopt the term *activation* from the spreading activation method[1] and we use it as a measure of a connectivity between source nodes (initial nodes defined by a user profile) and a target node (a Web API candidate). We use flow networks as an underlying concept for evaluation of the maximum activation in the graph. We implement the method as a Gephi plugin,[2] and we evaluate it on several experiments showing that the method gives better results over traditional popularity-based recommendations.

The remainder of this paper is structured as follows. Section 2 describes the underlying Linked Web APIs dataset and Section 3 describes the maximum activation method, its definitions and the algorithm. Section 4 describes several experiments from running the method on the Linked Web APIs dataset and a case study that shows how a developer can use the method when creating a mashup with various Web APIs. Section 5 describes the related work that also includes information on how the method compares to the spreading activation method. Finally, Section 6 concludes the paper and describes our future work.

2 Linked Web APIs

Figure 1 shows an extract of the Linked Web APIs dataset of the ProgrammableWeb, currently the largest directory of Web APIs. The Linked Web APIs dataset represents the whole directory as an RDF graph with over 300K RDF triples. To build the dataset we gathered data about Web APIs, mashups, their categories, developer profiles, and relationships among Web APIs and

[2] `https://github.com/jaroslav-kuchar/Maximum-Activation-Method`

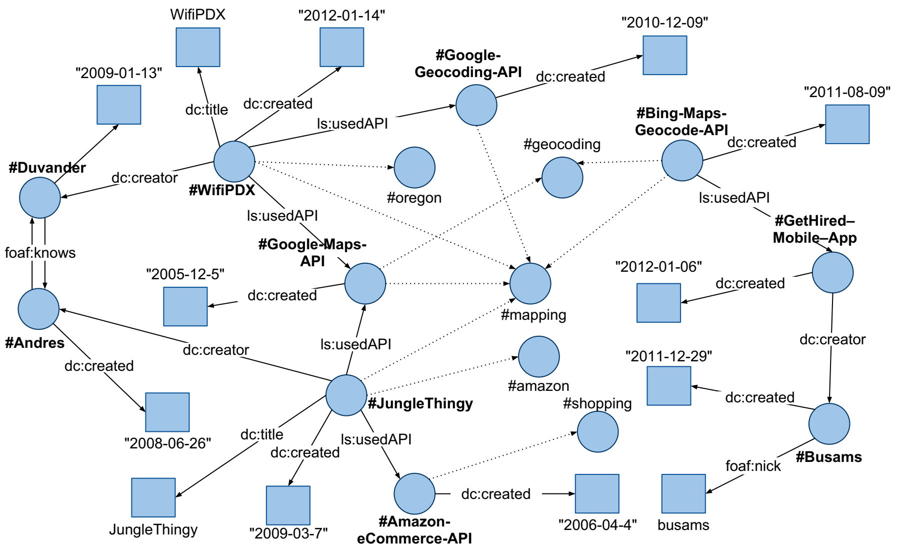


Fig. 1. Excerpt from the Linked Web APIs dataset

mashups that use them, relationships among Web APIs, mashups and developers who developed them, and relationships among Web APIs, mashups and categories. Moreover, we also capture the time of the Web APIs and mashups when they appeared in the ProgrammableWeb directory for the first time.

Note that there are other information in the ProgrammableWeb that we could use to make the Linked Web APIs richer such as various technical information about protocols and data formats. Also, we could better associate the data with other datasets in the Link Data cloud and publish it to the Linked Data community. Although we plan to do this in our future work, the Linked Web APIs dataset that we present here already provides the sufficient information for our Web API selection method.

The Linked Web API dataset uses several well-known ontologies. Concepts from FOAF[3] ontology (prefix `foaf`) represent mashup developers as *foaf:person* concepts with their social links, concepts from the WSMO-lite [15] ontology (prefix *wl*) represent Web APIs as *wl:service* concepts and their functional category descriptions. We also use the Dublin Core[4] vocabulary (prefix `dc`) for properties such as *title*, *creator* and *date*, and the SAWSDL [12] property *sawsdl:modelReference*. Further, we create new concepts and properties for which we use the `ls` prefix. We define the *ls:mashup* concept that represents a mashup and the *ls:category* concept that represents a functional Web API/mashup category. There are following types of edges in the Linked Web APIs:

3 http://xmlns.com/foaf/0.1/
4 http://dublincore.org/documents/dcmi-terms/

1. *User—User*: an edge between two user nodes represented with the *foaf:knows* property indicating a social link.
2. *User—Mashup*: an edge between a user and a mashup represented with the *dc:creator* property.
3. *Mashup—API*: an edge between a mashup and an API represented with the *ls:usedAPI* property.
4. *Mashup—Category*, and *API—Category*: an edge between a mashup/API and a category represented with the *sawsdl:modelReference* property.

3 Maximum Activation Method

3.1 Definitions

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{I})$ be a graph representing Linked Web APIs where $\mathcal{V}$ is a set of nodes, $\mathcal{E}$ is a set of edges and $\mathcal{I} : \mathcal{E} \to \mathbb{N}$ is a capacity function which associates a capacity of an edge with a natural number. A node in $\mathcal{V}$ can represent an API described by the *wl:Service* concept, a mashup described by the *ls:Mashup* concept, a user described by the *foaf:Person* concept or a category described by the *ls:Category* concept. An edge $e \in \mathcal{E}$ represents a mutual (bidirectional) relationship between two nodes as follows: for a property in the Linked Web APIs dataset we create an *inverse property* such that when (o_1, p, o_2) is a triple where o_1, o_2 correspond to nodes in $\mathcal{V}$ and p corresponds to an edge in $\mathcal{E}$, we create a new triple (o_1, p^{-1}, o_2) where p^{-1} is an inverse property to p. See Section 3.3 for additional details.

Let $P = \{p_1, p_2, ..., p_n\}$, $p_i \in \mathcal{V}$ be a set of nodes that represent a user profile. The nodes in P may represent the user himself, nodes that the user likes or knows or has any other explicit or implicit relationships with. Further, let $W = \{w_1, w_2, ..., w_m\}$, $w_i \in \mathcal{V}$ be a set of nodes that represent a user request as Web APIs candidates. The Maximum Activation method then calculates a maximum activation a_i for each Web API candidate $w_i \in \mathcal{W}$. The higher number of the maximum activation denotes a Web API candidate with a higher rank, that is the preferred Web API candidate over a Web API candidate with a lower maximum activation.

We denote an activation that can be sent between two nodes linked with an edge e as a natural number $i(e) \in \mathbb{N}$. The activation sent through an edge cannot exceed the capacity of the edge defined by the capacity function

$$\mathcal{I}(e_{i,t}) = S(e_i) * \mathcal{A}(e_{i,t}) \tag{1}$$

where $S(e_i) \in \{x \in \mathbb{N} | 0 \leq x \leq 100\}$ is a user preference function that defines an importance of the edge e_i (i.e., how the user sees an importance of semantics represented by the edge) and $A(e_{i,t})$ is the *exponential ageing function.* An importance $S(e_i) < 50$ indicates that the user does not prefer the edge's semantics, an importance $S(e_i) > 50$ indicates the the user prefers the edge's semantics and the importance $S(e_i) = 50$ indicates a neutral value. A user may chose an arbitrary number of edges for which he/she defines preferences. Edges for which the user does not define any preferences have a default preference 50.

Further, we define the exponential ageing function as

$$\mathcal{A}(e_{i,t}) = \mathcal{A}(e_{i,t_o}) * e^{-\lambda t} \tag{2}$$

where $\mathcal{A}(e_{i,t})$ is an age of the edge e_i at time t, $\mathcal{A}(e_{i,t_o})$ is the initial age of the edge e_i at the time the edge appeared in the graph $\mathcal{G}$ (i.e., values of *dc:created* property) and λ is an *ageing constant.* The ageing constant allows to configure an acceleration of the ageing process. Since our method gives better results for better connected nodes in the graph, the ageing function allows to control an advantage of "older" nodes that are likely to have more links when compared to "'younger" ones (see Section 4.1 for discussions on how we setup the ageing constant and Section 4.3 and 4.4 for differences in results with and without the ageing function applied).

Note that we currently only apply the ageing function to edges that are linked with nodes representing Web APIs and mashups. In other words, we use a creation date of a Web API or a mashup to evaluate the ageing function of any edge that links with the Web API or the mashup respectively. We assume that the Web API or the mashup was created at the same time along with all its edges that connect it to other nodes in the graph. For all other edges it holds that $\mathcal{A}(e_{i,t}) = 1$.

3.2 Algorithm

We calculate the Maximum Activation according to the following algorithm.

Inputs:
- Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{I})$ constructed from the Web Linked APIs dataset.
- A user profile $P = \{p_1, p_2, ..., p_n\}$.
- Web API candidates $W = \{w_1, w_2, ..., w_m\}$.
- A user preference function $S(e_i)$.

Output:
- A set of maximum activations $\{a_i\}$ evaluated for each $w_i \in W$.

Uses:
- A set $C = \{e_1, e_2, ..., e_k\}$, $e_i \in \mathcal{E}$.
- A function *FF* that represents the Ford-Fulkerson algorithm [8].

Algorithm:
```
1: // create a virtual source node $p'$
2: add node $p'$ to $\mathcal{V}$
3: for all $p_i \in P$ do
4:     add edge $e(p', p_i)$ to $\mathcal{E}$, $S(e) \leftarrow 100000$, $\mathcal{A}(e) \leftarrow 1$
5: end for
6: // calculate a maximum activation $a_i$ from
7: // a virtual node $p'$ to every Web API candidate $w_i$
8: for all $w_i \in W$ do
9:     $C \leftarrow FF(p', w_i, \mathcal{G})$
10:    $a_i \leftarrow \sum_{e_i \in C}(\mathcal{I}(e_i))$
11: end for
```

In lines 2–5, the algorithm first creates a virtual node representing a single source node with links connecting the virtual node and all other nodes from the user profile. Any edge that connects the virtual node with any other node in the graph has a capacity set to a very large value so that the edge does not constrain the maximum activation. In lines 8–11, the algorithm finds a maximum activation for each Web API candidate w_i from the virtual node p'. For this we use the Ford-Fulkerson algorithm to find a maximum activation from the *source node* (i.e., the virtual node) to the *target node* (i.e., a Web API candidate). We do not formaly describe the *FF* algorithm here, however, for the purposes of later discussion in Section 3.3 we provide its brief description: the *FF* algorithm first sets the initial activation for each edge to 0 and tries to find an *improving path* on which it is possible to increase the activation by a minimum value greater then 0. If such path is found, the algorithm increases activations on every edge on the path and tries to find another path. When no more path is found, the algorithm ends. The result of *FF* is the set C that contains every last edge from all paths from the source towards the target when an improving path is not possible to find. The maximum activation is the sum of all activations on edges from C. In line 10, the algorithm finally calculates the maximum activation as a sum of all activations of edges in C.

3.3 Discussion

Meaning of Maximum Activation Value. As we noted earlier, we interpret the maximum activation of the graph as a measure that indicates how well the source nodes are connected with the target. In general, the more improving paths exist between the source and the target, the higher maximum activation we can get. However, the value of the maximum activation is also dependent on constraints and the creation time of Web APIs and mashups along the improving paths when the ageing function is applied.

Maximum Activation and Edges in C. The edges in C are constraining the maximum activation which means that if capacities of such edges increase, the maximum activation can be increased. Note, however, that we assign capacities based on semantics of egdes thus by changing a capacity on an edge in C, we also change capacities on other edges not in C. Running the algorithm again on the graph with new capacities will lead to a different set C and different maximum activation. In other words, it does not hold that increasing a capacity on any edge in C will lead to a higher maximum activation. This also means that maximum activation that our algorithm evaluates has a global meaning while activations on individual edges do not have any meaning. Defining capacities for individual edges is the subject of our future work.

Graph $\mathcal{G}$ Construction. When we construct the graph $\mathcal{G}$ from the Linked Web APIs dataset, for every predicate we create a bidirectional edge. A graph with bidirectional edges provides a richer dataset for maximum activation evaluation. A large graph with unidirectional edges may contain many dead end paths that may limit the number of improving paths that the algorithm would be able to find from the source to the target nodes. Evaluation of maximum activation on such graph would not provide many interesting results.

4 Experiments

In this section we present several experiments and their results[5] that use maximum activation for the Web APIs selection.

For our experiments we use the full Linked Web APIs dataset. The dataset contains all user profiles for users that created at least one mashup. We also extracted profiles on all categories, tags, mashups and Web APIs. The snapshot we use covers the period from the first published API description in June 2005, till May 18th, 2012. The snapshot includes 5 988 APIs, 6 628 mashups and 2 335 user profiles. In the experiments we addressed following questions:

– *What is the impact of user preference function on results of the maximum activation?*
– *How does the ageing factor influence the maximum activation?*
– *How can the popularity of an API evolve over time?*
– *How to make the process of building a mashup more personalised and contextualised?*

4.1 Setting the Ageing Constant

We experimentally set the ageing constant to a value $\lambda = 0.1$ and the age period to one week ($t = week$). Our graph contains data since June 2005, that is approximately 360 weeks. Figure 2 depicts an effect on ageing function for different λ. Note that the higher the constant is, the algorithm promotes the more recently added APIs and Mashups.

4.2 Impact of the User Preference Function

The user preference function defines an importance of the edge, that is how the user sees the importance of semantics represented by the edge. For example, the user can give a higher importance to edges representing a friendship (*foaf:knows*) than to edges between mashups and Web APIs (*ls:usedAPI*). The importance values, along with the chosen edge ageing constant λ, are used to compute the total capacity of an edge (see definition (1)). To study the influence of an importance value on a single edge, we were gradually increasing the value from 0 to 100 by a step of 5 and fixed importance values of all other edges to 50. We run this experiment for 3 different well-known APIs, namely Google Maps, Bing Maps and Yahoo Maps.

Figure 3 shows the experiment results: the importance value on the edge *API–Mashup* (Fig.3(b)) and *Mashup–User* (Fig.3(g)) does not have influence

[5] Full results of the experiments are available at `http://goo.gl/GKIbo`

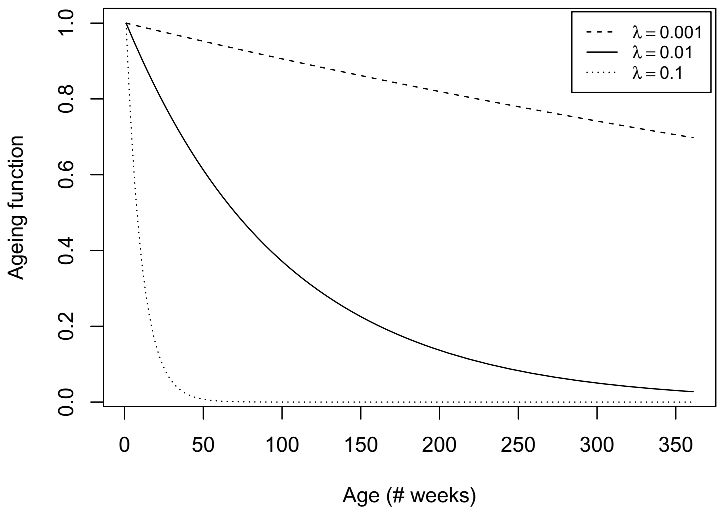


Fig. 2. Ageing function

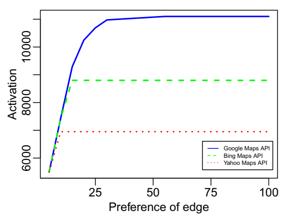
on the maximum activation. Slight influence has the importance value on edges *Mashup–Category* (Fig.3(f)), *User–Mashup* (Fig.3(h)) and *User–User* (Fig.3(i)). Fig.3(a) further shows that different importance values have various ranges of influence: the importance value for the *API–Category* has influence in a range $0-5$ for Yahoo Maps API, $0-10$ for Bing Maps API, and $0-15$ for Google Maps API, while higher importance values do not have any influence as the maximum activation is limited by the capacities of other types of edges.

4.3 Impact of the Ageing Constant λ

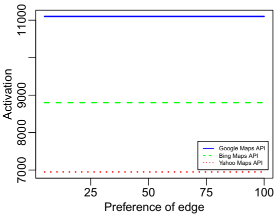
The ageing constant λ is a configurable parameter which influences the value of assigned edge capacity. The higher the λ is, the more recent edges will be preferred – that is, the older edges will have a lower capacity. In different datasets edges can occur more or less frequently therefore appropriate value for the λ should be set. Setting high λ in datasets where the edges occur less frequently may lead to very low edge capacities and consequently to the low activation value. In other words, the ageing constant λ makes the selection method more adaptable to different datasets.

For this experiment we chose a random user Dave Schappell[6] and we calculated the maximum activation for each API candidate in the "mapping" category. We evaluated the results in the period from 1st of June 2009 (shortly after the user registered his profile) till 1st of June 2012 with a period of age set to 1 week. We set the ageing constant λ to values 0.01 and 0.1. By setting the ageing

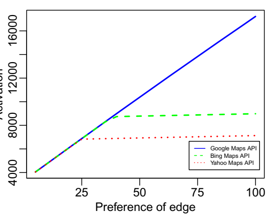
[6] http://www.programmableweb.com/profile/daveschappell



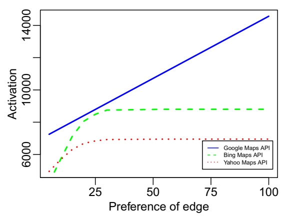
(a) API – Category



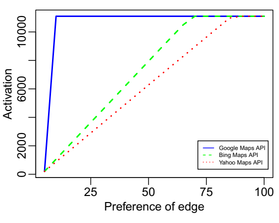
(b) API – Mashup



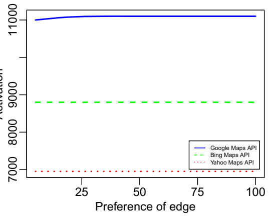
(c) Category – API



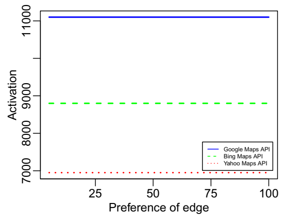
(d) Category – Mashup



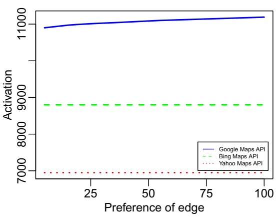
(e) Mashup – API



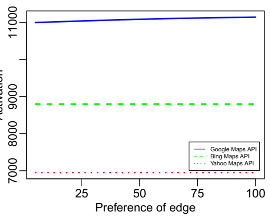
(f) Mashup – Category



(g) Mashup – User



(h) User – Mashup



(i) User – User

Fig. 3. Impact of Importance values

constant we are able to accelerate the ageing process, that is we get a lower capacity on older edges. Fig.2 shows, setting the ageing constant to 0.1 we get higher maximum activation for edges that appeared in the last 50 weeks, and setting it to 0.01 in the last 350 weeks.

Table 1 shows the configuration of importance values for various types of edges for this experiment and Table 2 and 3 shows the results of this experiment for λ set to 0.01 and 0.1 respectively. In these tables, the“PW rank” column shows a popularity-based ranking used by the ProgrammableWeb which is only based on

Table 1. Importance Value Configuration

Edge name	Importance value	Edge name	Importance value
API–Category	50	Mashup–Category	70
API–Mashup	50	Mashup–User	50
Category–API	70	User–Mashup	90
Category–Mashup	20	User–User	90
Mashup–API	70	/	/

Table 2. Summarised ranking results with λ=0.01

Node ID	API name	Date created	Max-Activation $\lambda = 0.01$ value	Max-Activation $\lambda = 0.01$ rank	PW rank
2053	Google Maps API	2005-12-05	**5559**	1	1
2041	Google Earth API	2008-06-01	1080	2	5
2057	Google Maps Data API	2009-05-20	1043	3	8
2052	Google Geocoding API	2010-12-09	1028	4	11
3032	Microsoft Bing Maps API	2009-06-09	853	5	2
2060	Google Maps Flash API	2008-05-17	792	6	6
5827	Yahoo Geocoding API	2006-02-14	715	7	4
5836	Yahoo Maps API	2005-11-19	707	8	3
493	Bing Maps API	2009-06-09	662	9	10
2070	Google Places API	2010-05-20	553	10	18

Table 3. Summarised ranking results with λ=0.1

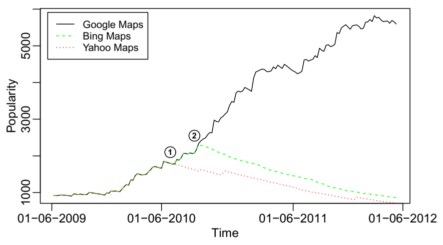
Node ID	API name	Date created	Max-Activation $\lambda = 0.1$ value	Max-Activation $\lambda = 0.1$ rank	PW rank
2053	Google Maps API	2005-12-05	**503**	1	1
5531	Waytag API	2012-04-27	210	2	230
4330	Scout for Apps API	2012-04-20	190	3	202
4535	Google Geocoding API	2010-12-09	184	4	11
3815	Pin Drop API	2012-03-27	135	5	191
5950	Zippopotamus API	2012-04-26	123	6	233
5825	Yahoo Geo Location API	2012-04-23	120	7	230
1836	FreeGeoIP API	2012-03-29	112	8	116
5156	Trillium Global Locator API	2012-04-18	111	9	109
1430	eCoComa Geo API	2012-05-15	108	10	108

a number of mashups used by an API. Google Maps API is the highest ranked API by our method (for both λ=0.01 and λ=0.1) and also is the highest ranked by the Programmable Web popularity-based method. For $\lambda = 0.01$, the method favors the recent APIs but also does not ignore APIs that were actively used in the past 350 months (approx. 7 years).

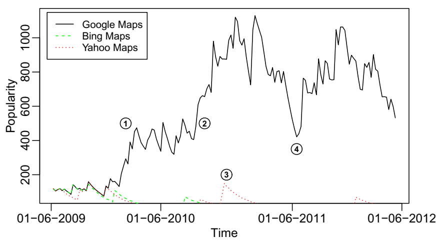
From the results in Table 3 it is possible to see that the ageing constant $\lambda = 0.1$ promotes newer APIs while at the same time it does not ignore the all-time popular APIs such as Google Maps API and Google Geocoding.

4.4 Popularity of APIs over Time

In this experiment we examine a popularity of 3 APIs from the "mapping" category for the user Dave Schappell in different points in time. We use the configuration in Table 1 and the ageing constant λ set to values 0.01 and 0.1.



(a) With ageing constant $\lambda = 0.01$



(b) With ageing constant $\lambda = 0.1$

Fig. 4. API popularity over Time

The results show that the Google Maps API has the highest popularity in both cases for the ageing constant set at 0.01 and 0.1. From Figure 4(a) we can see that the popularity of Yahoo Maps API and Bing Maps API follows the popularity of the Google Maps API until the time marked with (1) and (2). After the times (1) and (2), a popularity of the two APIs starts to fall. Around December 2010 and January 2012 the popularity of Yahoo Maps API experienced minimal activation growth due to several new mashups that were created and used this API.

Figure 4(b) shows a popularity of the three APIs with a more strict edge ageing. After the first half year, when the popularity of the 3 APIs is nearly the same, the popularity of the Google Maps API is starting to increase until the time marked with (1) and stays at this level until the time marked with (2). Between the times (2) and (4) Google Maps API gained a popularity up to maximum activation of 1 129, however, it also started to lose some activation due to a less number of mashups that were using this API. On the other hand, popularity of the Yahoo Maps API increased around December 2010 (3) due to its more intensive usage. As we can see, in certain cases, by using the ageing function we can get better results than the PW'S popularity-based ranking.

4.5 Case Study

In this section we present a case study for personalised API selection to illustrate capabilities of our maximum activation method. We have a developer who wants to improve tourists' experience in New York, USA by creating the Visitor Mashup. The Visitor Mashup should aggregate information about different events and information about restaurants in the city and in the area of New York. Information about various points of events and restaurants should be layered on the map and dynamically updated when tourists change their locations and new events and restaurants become available.

Developer starts the process of building the Visitor Mashup by identifying groups of relevant APIs. As he progresses and selects APIs, the ranking process becomes more personalised and contextualised. The process of creating the

Visitor Mashup is described by following steps when in each step the developer selects one API:

– **Maps API.** Developer builds his profile adding "maps" and "location" categories to it. He assigns a high importance value to the "API–Category". Table 4 shows the highest ranked results: Google Maps, Microsoft Bing Maps and Yahoo Maps. The developer decides to select the Google Maps API.

Table 4. Summarised ranking results for Maps API

Node ID	API name	Date created	Max-Activation λ not set value	Max-Activation λ not set rank	Max-Activation $\lambda = 0.01$ value	Max-Activation $\lambda = 0.01$ rank	PW rank
2053	Google Maps API	2005-12-05	**13720**	1	**6509**	1	1
3032	Bing Maps API	2009-06-09	3720	2	238	2	10
5836	Yahoo Maps API	2005-11-19	2980	3	172	3	3

– **Events API.** The developer further searches for events API by updating his profile with "events" category, adding "Google Maps API" and preserving "maps" and "location" categories. Further, he increases an importance value of the "Mashup–API". Table 5 shows highest ranked results: Seatwave, Eventful and Upcoming.rg. The developer selects Seatwave API.

– **Restaurant API.** The developer searches restaurants API by adding "'food", "restaurants" and "menus" categories to his profile. This time the developer decides to use his social links and to look for APIs used by his friends developers that he adds to his profile. Table 6 shows the highest ranked APIs SinglePlatform, Menu Mania and BooRah. The developer selects SinglePlatform API for restaurant information and recommendations.

Table 5. Summarised ranking results for Events API

Node ID	API name	Date created	Max-Activation λ not set value	Max-Activation λ not set rank	Max-Activation $\lambda = 0.01$ value	Max-Activation $\lambda = 0.01$ rank	PW rank
4348	Seatwave API	2012-02-28	940	3	**842**	**1**	4
1578	Eventful API	2005-10-31	3930	1	710	2	1
5371	Upcoming.rg API	2005-11-19	3220	2	411	3	2

Table 6. Summarised ranking results for Restaurant API

Node ID	API name	Date created	Max-Activation λ not set		Max-Activation $\lambda = 0.01$		PW rank
			value	rank	value	rank	
4522	SinglePlatform API	2012-01-30	150	2	**125**	1	6
2980	Menu Mania API	2009-12-05	**220**	1	65	2	1
611	BooRah API	2008-10-31	120	3	30	3	3

5 Related Work

Graph-based representation of services is a relatively new approach. The authors in [2] propose service selection based on previously captured user preferences using the "Follow the Leader" model. In [14] the authors construct collaboration network of APIs and propose a social API Rank based on the past APIs' utilisations. Other approaches that rank services based on results from social network-based analyses in social API networks can be found in [17] and [13].

A particular method that relates to our work is the already mentioned spreading activation. It is a graph-based technique, originally proposed as a model of the way how associative reasoning works in the human mind [4]. The spreading activation requires directed semantic network, e.g. an RDF graph [5,9,7]. The inputs of the basic spreading activation algorithm are number of nodes with an initial activation which represent a query or interests of a user. In sequence of iterations initial (active) nodes pass some activation to connected nodes, usually with some weighting of connections determining how much spread gets to each. This is then iterated until some termination condition is met. The termination conditions is usually represented as a maximum number of activated nodes or a number of iterations. After the algorithm terminates, activated nodes represent a similar nodes to the initial set of nodes.

Compared to our maximum activation method, the spreading activation does not guarantee an activation of a particular node while our method always assigns an activation if there exists an improving path between source and target nodes. Although there exist constrained spreading activation methods which utilise semantics of edges [6], no version of the spreading activation takes into account the "age" of edges as our method does. The maximum activation is better suited for the Web API selection mainly due to following reasons: 1) it is not known at which nodes the spreading activation terminates while the Web API selection problem uses Web API candidates as an input (target nodes), 2) the spreading activation has a local meaning of activations that indicates a measure that can be used for recommendations on data whereas maximum activation uses the value as a global measure of connectivity from source to target nodes.

There are other works in the area of Web Service discovery and selection including QoS selection [10,18], collaborative and content-based filtering methods [3,20,11,19] which are less relevant.

6 Conclusion and Future Work

A popularity and a growing number of Web APIs and mashups require new methods that users can use for more precise selection of Web APIs. Current approaches for searching and selecting Web APIs utilize rankings based on Web APIs popularity either explicitly expressed by users or a number of Web APIs used in mashups. Such metrics works well for the large, widely-known and well-established APIs such as Google APIs, however, they impede adoption of more recent, newly created APIs. In order to address this problem we proposed a novel activation-based Web API selection method which takes into account a user profile and user's preferences, temporal aspects (the creation time of Web APIs and mashups) and social links between users. While existing popularity-based rankings use a single-dimensional ranking criteria (i.e., a number of APIs used in mashups), our method uses multi-dimensional ranking criteria and with help of graph analysis methods it provides more precise results. The method requires a set of Web API candidates, a user profile and evaluates a ranking for all Web API candidates for the given user profile. The Web API candidates may result from a service discovery task that usually evaluates a match based on a coarse-grained search request. Service discovery requests may be represented as a functional category, for example, the discovery returns all services in the same category such as a mapping category.

In our future work we want to extend the method so that we can assign capacities to individual edges. In cooperation with ProgrammableWeb.com, we also plan to improve the Linked Web APIs dataset and eventually make it available in the Linked Data cloud. We want to enrich this dataset with user profiles from traditional social networks. We also plan to incorporate to our method various social network analysis metrics evaluated on the Linked Web APIs dataset. Last but not least we want to evaluate the method on datasets from the Linked Data cloud.

Acknowledgement. This work was supported by the Czech Technical University grant number SGS12/093/OHK3/1T/18 and the In-Coming Scholarship with application number 51100570 provided by the International Visegrad Fund. We also thank to ProgrammableWeb.com for supporting this research.

References

1. Akim, N.M., et al.: Spreading activation for web scale reasoning: Promise and problems. In: WebSci. (2011)
2. Al-Sharawneh, J., Williams, M.-A.: A social network approach in semantic web services selection using follow the leader behavior. In: 13th Enterprise Distributed Object Computing Conference Workshops, EDOCW 2009, pp. 310–319 (September 2009)
3. Lecue, F.: Combining collaborative filtering and semantic content-based approaches to recommend web services. In: 2010 IEEE Fourth International Conference on Semantic Computing (ICSC), pp. 200–205 (September 2010)

4. Anderson, J.R.: A spreading activation theory of memory. Journal of Verbal Learning and Verbal Behavior 22, 261–295 (1983)
5. Choudhury, S., Breslin, J.G., Passant, A.: Enrichment and Ranking of the YouTube Tag Space and Integration with the Linked Data Cloud. In: Bernstein, A., Karger, D.R., Heath, T., Feigenbaum, L., Maynard, D., Motta, E., Thirunarayan, K. (eds.) ISWC 2009. LNCS, vol. 5823, pp. 747–762. Springer, Heidelberg (2009)
6. Crestani, F.: Application of spreading activation techniques in information retrieval. Artificial Intelligence Review 11, 453–482 (1997)
7. Dix, A., et al.: Spreading activation over ontology-based resources: from personal context to web scale reasoning. International Journal of Semantic Computing 4(1), 59 (2010)
8. Ford, L.R., Fulkerson, D.R.: Maximal flow through a network. Canadian Journal of Mathematics 8, 399–404 (1956)
9. Freitas, A., et al.: Querying linked data using semantic relatedness: a vocabulary independent approach. In: Natural Language Processing and Information Systems, pp. 40–51 (2011)
10. Godse, M., Bellur, U., Sonar, R.: Automating qos based service selection. In: 2010 IEEE International Conference on Web Services (ICWS), pp. 534–541 (July 2010)
11. Jiang, Y., Liu, J., Tang, M., Liu, X.: An effective web service recommendation method based on personalized collaborative filtering. In: 2011 IEEE International Conference on Web Services (ICWS), pp. 211–218 (July 2011)
12. Kopecky, J., Vitvar, T., Bournez, C., Farrell, J.: Sawsdl: Semantic annotations for wsdl and xml schema. IEEE Internet Computing 11(6), 60–67 (2007)
13. Shafiq, M., Alhajj, R., Rokne, J.: On the social aspects of personalized ranking for web services. In: 2011 IEEE 13th International Conference on High Performance Computing and Communications (HPCC), pp. 86–93 (September 2011)
14. Torres, R., Tapia, B., Astudillo, H.: Improving web api discovery by leveraging social information. In: 2011 IEEE International Conference on Web Services (ICWS), pp. 744–745 (July 2011)
15. Vitvar, T., Kopecký, J., Viskova, J., Fensel, D.: WSMO-Lite Annotations for Web Services. In: Bechhofer, S., Hauswirth, M., Hoffmann, J., Koubarakis, M. (eds.) ESWC 2008. LNCS, vol. 5021, pp. 674–689. Springer, Heidelberg (2008)
16. Vitvar, T., Vinoski, S., Pautaso, C.: Programmatic interfaces for web applications, guest introduction. IEEE Internet Computing (to appear, July/August 2012)
17. Wang, S., Zhu, X., Zhang, H.: Web service selection in trustworthy collaboration network. In: 2011 IEEE 8th International Conference on e-Business Engineering (ICEBE), pp. 153–160 (October 2011)
18. Yau, S., Yin, Y.: Qos-based service ranking and selection for service-based systems. In: 2011 IEEE International Conference on Services Computing (SCC), pp. 56–63 (July 2011)
19. Zhang, Q., Ding, C., Chi, C.-H.: Collaborative filtering based service ranking using invocation histories. In: 2011 IEEE International Conference on Web Services (ICWS), pp. 195–202 (July 2011)
20. Zheng, Z., Ma, H., Lyu, M., King, I.: Wsrec: A collaborative filtering based web service recommender system. In: IEEE International Conference on Web Services, ICWS 2009, pp. 437–444 (July 2009)

Instance-Based Matching of Large Ontologies Using Locality-Sensitive Hashing

Songyun Duan, Achille Fokoue, Oktie Hassanzadeh, Anastasios Kementsietsidis, Kavitha Srinivas, and Michael J. Ward

IBM T.J. Watson Research,
19 Skyline Drive, Hawthorne, NY 10532
{sduan,achille,hassanzadeh,akement,ksrinivs,MichaelJWard}@us.ibm.com

Abstract. In this paper, we describe a mechanism for ontology alignment using instance based matching of types (or classes). Instance-based matching is known to be a useful technique for matching ontologies that have different names and different structures. A key problem in instance matching of types, however, is scaling the matching algorithm to (a) handle types with a large number of instances, and (b) efficiently match a large number of type pairs. We propose the use of state-of-the art locality-sensitive hashing (LSH) techniques to vastly improve the scalability of instance matching across multiple types. We show the feasibility of our approach with DBpedia and Freebase, two different type systems with hundreds and thousands of types, respectively. We describe how these techniques can be used to estimate containment or equivalence relations between two type systems, and we compare two different LSH techniques for computing instance similarity.

Keywords: Ontology Alignment, Schema Matching, Linked Data, Semantic Web.

1 Introduction

Ontology (or schema) matching is a well-studied problem in the literature that has received considerable attention over the last decade, as is clearly evident from the large number of papers published over the years [30,15,27,5,17,21,19, and many others]. In these works, the predominant approach to matching exploits purely schema-related information, i.e., labels or structural information about the type hierarchy. This *schema-based* approach to schema matching is a practical starting point that proves adequate in a number of applications. However, schema-based matchers have their limitations, especially in situations where schema elements have obscured names [27]. This observation gave rise to a class of *instance-based* matchers [29,16,4,18,6] in which the instance data are consulted as well in order to determine the schema mappings.

For both classes of matchers, the focus of most of these past works has been on achieving high precision and/or recall. While these are important evaluation metrics to illustrate the correctness of the developed techniques, a metric that is often ignored in the evaluation is *scalability*. With the rapid rise in the size and number of data sources on the web, effective schema matching techniques must be developed that work at web scale. One only has to look at the Linked Open Data cloud [7] to be instantly exposed to approximately 300 sources with thousands of (RDF) types (or classes), with new sources of types added constantly. Data that reside in different sources in the web is clearly associated, but discovering these relationships can only be achieved if we are able to deduce which types in the various sources are related. As a simple example, consider an entity like the city of Boston. It is not hard to see that information for this entity can be found in the following datasets: (a) the DBpedia entity for Boston has the type `http://dbpedia.org/ontology/City`; (b) the Freebase entity for Boston has the type `http://rdf.freebase.com/rdf/location/citytown` or `http://rdf.freebase.com/location/location`; (c) in RDFa, using the schema.org vocabulary, it has type `http://schema.org/Place` or `http://schema.org/City`; and (d) the GeoNames entity has the type `http://www.geonames.org/ontology#P`.

Clearly, we would like to be able to create matchings between all these types with the different (often obscure) names in the different sources. Given that instance-based approaches are more appropriate to deal with the differences in schema vocabularies, it seems appropriate to consider such techniques in this context. However, scalability is a key problem in applying these techniques to web scale. To see why, consider a simple setting in which we have n types in one data source, m types in another, and we assume that we have l instances for each of these types. Then existing approaches would require $n \times m$ type comparisons, where each type comparison requires at least $O(l)$ instance comparison operations. Clearly, this is not scalable for most realistic usage scenarios.

In this paper, we focus on the problem of scaling instance-based ontology alignment using locality-sensitive hashing (LSH) [32] techniques drawn from data mining. Specifically, we show how one can use LSH techniques such as MinHash [10] and random projection (a.k.a. *random hyperplane* or *RHP*) to estimate instance similarity [13] and hence infer type similarity. To compute instance similarity between two types, we first need to define the granularity with which an instance is defined for the purposes of similarity analysis. Whole instances frequently do not match between different type systems because of slight differences in representing instances as strings, e.g., "Bank of America" versus "Bank of America, Ltd". As a result, we compute instance similarity using tokenized strings to reduce the probability of misses due to string differences.

Instead of computing pairwise Jaccard similarity for all pairs of instance sets in the two type systems, we use the MinHash technique to efficiently estimate Jaccard similarity with a very low error rate while (a) compressing the instance data to a fixed size for comparison, such that l instances can be succinctly represented by a set of k hashes, where k is a small fixed number such as 500; and (b) eliminating most of the irrelevant $n \times m$ type comparisons efficiently while providing mathematical guarantees about false negatives (the algorithm cannot introduce false positives since it only prunes potentially irrelevant comparisons, *i.e.*, those with similarity measures below a given threshold).

While set similarity is one metric for measuring instance similarity, it can be unduly influenced by terms that occur very frequently across all types. We therefore also estimate cosine similarity of term frequency vectors for instances of each type, where the term frequencies are weighted by the $tf * idf$ measure. In the context of instance similarity computations, $tf * idf$ is a weight that reflects the degree to which a term appears in a particular type, compared to its occurrence in all types. This measure then corrects for possible biases in the similarity computation due to frequently occurring words.

As in the case of Jaccard similarity, we estimate cosine similarity on all types using LSH techniques, because an all-to-all cosine similarity computation is not feasible in practice. Specifically, we use the random projection method for estimating cosine similarity between term frequency vectors for all candidate type pairs (*i.e.*, pairs with an expected similarity above a given threshold). The core idea behind the random projection method relies on choosing a set of k random hyperplanes to hash the input vectors [13], thus once again allowing comparisons of k hashes.

Both cosine and Jaccard measures of instance similarity can be affected negatively when the sets of instances between two types being compared have very disparate sizes. For instance, if type A has 10 instances, and type B has 100 instances, the maximum Jaccard similarity one can get is 10/100 or 0.1. We measure containment between types as well as their similarity to determine if the relationship between the two types reflects equivalence or containment.

Our contributions in this paper are: (a) we describe the use of LSH techniques for the efficient computation of instance-based similarity across disparate ontology types or schema elements, (b) we show how these techniques can be used to estimate containment or equivalence relations between two type systems, (c) we compare Jaccard similarity and cosine similarity, to correct for possible biases in estimation of similarity due to frequently occurring words, and (d) we evaluate the utility of this approach with Freebase and DBpedia, which are two large linked open datasets with different type systems.

2 Preliminaries: Locality Sensitive Hashing (LSH)

When comparing large numbers of types based on the similarity of their instances, there are two primary scalability hurdles:

- First, to compare two types with L instances each regardless of similarity metric, at least $O(L)$ operations are required. For large values of L, this repeated cost becomes the performance bottleneck.
- Second, to compare all pairs of types requires a quadratic computational cost; but in practice, only pairs of types with a high enough similarity are of interest. How can we save computation by pruning out those type pairs with low similarity?

2.1 Reducing Comparison Cost for One Type Pair

Locality Sensitive Hashing (LSH) [32] addresses the first scalability hurdle by approximating the similarity in the following way.

Let $\mathcal{U}$ be a set of objects, a similarity measure **sim** is a function from $\mathcal{U}^2$ to the interval of real numbers $[0, 1]$ such that, for u and v in $\mathcal{U}$, $\mathbf{sim}(u, v)$ indicates the relative similarity between u and v. For example, $\mathbf{sim}(u, v) = 0$ indicates no similarity at all between u and v, whereas $\mathbf{sim}(u, v) = 1$ corresponds to perfect match between u and v.

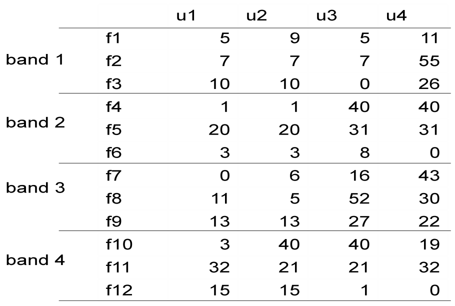
Let **sim** be a similarity measure defined in $\mathcal{U}^2$. A family $\mathcal{F}$ of hash functions from $\mathcal{U}$ to the set $\mathbb{Z}$ of integers is **sim**-*sensitive* iff., for any pair $(u, v) \in \mathcal{U}^2$, the probability $\mathbf{Pr}(f(u) = f(v))$ that a randomly chosen hash function f of $\mathcal{F}$ hashes u and v to the same value is equal to the similarity **sim** of u and v, that is, $\mathbf{Pr}(f(u) = f(v)) = \mathbf{sim}(u, v)$.

The key idea in LSH is to estimate the similarity $\mathbf{sim}(u, v)$ between two elements u and v more efficiently by randomly sampling hash functions from a hash family $\mathcal{F}$ to estimate the proportion of functions f such that $f(u) = f(v)$. From the sampling theory, we know that the number of functions, denoted as n, can be relatively small with a relatively small sampling error (*e.g.*, for $n = 500$, the maximum sampling error is about $\pm 4.5\%$ with 95% confidence interval). Hence, the similarity between u and v can be estimated based on a small number of functions from the hash family.

2.2 Avoiding Quadratic Comparisons

To address the second hurdle of avoiding the quadratic complexity associated with comparing all pairs of types, we describe another well known technique, called *banding*, that can help efficiently select pairs of types whose similarities are likely to be above a given threshold.

Let $(f_1, \ldots, f_n)$ be a list of n independent functions from a **sim**-*sensitive* hash family $\mathcal{F}$. The signature matrix, denoted $(f_1, \ldots, f_n)$-$sigM(\mathcal{U})$, is a matrix of n rows and $|\mathcal{U}|$ columns whose j^{th} column contains the signature, $(f_1, \ldots, f_n)$-$sig(u_j)$, of the j^{th} element u_j of $\mathcal{U}$. The cell $(f_1, \ldots, f_n)$-$sigM(\mathcal{U})[i, j]$ at row i



		u1	u2	u3	u4
	f1	5	9	5	11
band 1	f2	7	7	7	55
	f3	10	10	0	26
	f4	1	1	40	40
band 2	f5	20	20	31	31
	f6	3	3	8	0
	f7	0	6	16	43
band 3	f8	11	5	52	30
	f9	13	13	27	22
	f10	3	40	40	19
band 4	f11	32	21	21	32
	f12	15	15	1	0

Fig. 1. Example of Signature Matrix and bands

and column j ($1 \leq i \leq n$ and $1 \leq j \leq |\mathcal{U}|$) is equal to $f_i(u_j)$. Figure 1 shows an example of such a matrix with $n = 12$ and $|\mathcal{U}| = 4$.

The randomly selected functions $(f_1, \ldots, f_n)$ can be grouped into b mutually disjoint bands (or groups), each containing r functions ($n = b \times r$) as illustrated in Figure 1, where $b = 4$ and $r = 3$. For two elements u_j and u_k in $\mathcal{U}$ ($1 \leq j < k \leq |\mathcal{U}|$), the pair (u_j, u_k) is considered a candidate pair iff. there is a band b_l ($1 \leq l \leq b$) such that, for each function f in this band b_l, $f(u_j) = f(u_k)$. In other words, (u_j, u_k) is a candidate pair iff. there is a band b_l such that the signatures of u_j and u_k in that band are equal. For example, in Figure 1, (u_1, u_2) is a candidate pair because their signatures have the same value in band 2, whereas (u_1, u_3) is not a candidate pair because their signatures are different in all 4 bands. Intuitively, pairs with high enough similarities are more likely to have their signatures agree in at least one band.

Formally, the probability that a pair (u, v) with similarity s is selected by this banding technique is $1 - (1 - s^r)^b$. Regardless of the value of r and b, the curve (see Figure 2) representing the probability that a pair (u, v) is considered a candidate as a function of $\mathbf{sim}(u, v)$ has a characteristic S-shape. This means, for a given similarity threshold and a given acceptable false negative rate *rate* (which means type pairs with similarity above this threshold are missing), r and b can be chosen so as to maximize the likelihood that the actual false negative rate remains below the given parameter *rate*.

In practice, for efficiency (*i.e.*, to avoid pairwise comparisons), in each band b_l, projections of signatures to b_l are hashed by a hash function h, and elements whose projected signatures on b_l hash to the same value are put in the same bucket. Assuming that the chances of collisions for h are negligible, two elements u and v will end up in the same bucket of a band b_l iff. their signatures agree in the band b_l. Finally, a pair (u, v) is considered a candidate iff. there is at least a band in which u and v are put in the same bucket.

3 Instance-Based Type Matching with LSH

For matching a finite set $\mathcal{T}$ of types based on their instances, we take an Information Retrieval (IR) approach to associate a list of terms $\mathbf{termlist}(t)$ to each type $t \in \mathcal{T}$. Conceptually, for a given type t, we build a document d_t by concatenating, for a given property (*e.g.*, *rdfs:label*), all its values for all instances of t. After applying standard IR processing to d_t (*e.g.*, tokenization, lowercasing, stemming, etc.), we obtain a list of terms $\mathbf{termlist}(t)$. For two types t and t', we use the similarity between $\mathbf{termlist}(t)$ and $\mathbf{termlist}(t')$ as a measure of the similarity between t and t'. We consider cosine (**cossim**) similarity and Jaccard (**jaccsim**) similarity, and their well known LSH equivalents using the random projection method and the MinHash method, respectively.

3.1 Cosine Similarity (Random Projection Method)

To measure the cosine similarity of two types, we weight each term $r \in \mathbf{termlist}(t)$ using the formula $\mathbf{termvec}(t)[r] = \mathbf{tf}(r, d_t) \times \mathbf{idf}(r)$, where $\mathbf{tf}(r, d_t)$,

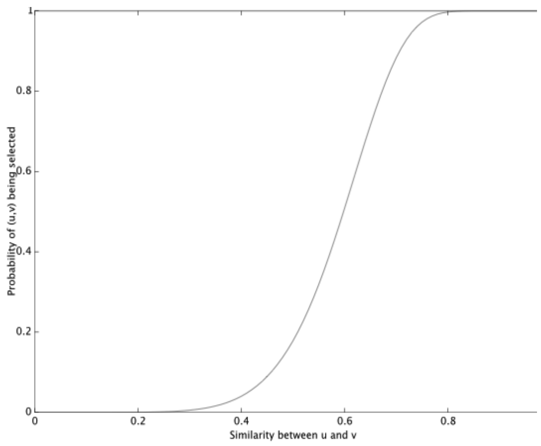


Fig. 2. Probability of a (u, v) selected as a function of $\mathbf{sim}(u, v)$ ($r = 7$ & $b = 25$)

the term frequency, is the number of occurrences of r in $\mathbf{termlist}(t)$, and $\mathbf{idf}(r)$, the inverse document frequency, is defined as $\mathbf{idf}(r) = log(|\mathcal{T}|/(1 + d))$ where d is the number of types that contain this term r. $\mathbf{idf}(r)$ measures how common the term r is in all types; a very common term (i.e., low $idf(r)$) is not informative for type matching. We then use the random projection method to estimate cosine similarity, which we give a brief introduction below.

If $\mathcal{U}$ is a vector space, a traditional metric used to measure the similarity between two vectors u and v in $\mathcal{U}$ is cosine similarity, denoted $\mathbf{cossim}(u, v)$ and defined as the cosine of the angle θ between u and v (see Figure 3). A closely related similarity to the cosine similarity between two vectors u and v, the angular similarity $\mathbf{angsim}(u, v)$, is defined as $\mathbf{angsim}(u, v) = \frac{\pi-\theta}{\pi}$, where θ is the angle between u and v ($0 \leq \theta \leq \pi$). $\mathbf{cossim}(u, v)$ is computed from $\mathbf{angsim}(u, v)$ as follows: $\mathbf{cossim}(u, v) = -cos(\pi \times \mathbf{angsim}(u, v))$.

For the ease of presentation, we describe how **angsim**-sensitive family of functions can be constructed. Given two vectors u and v, let P denote the plane they form, presented in Figure 3. Consider a hyperplane H, which can be characterized by one of its normal vectors n. H intersects P in a line. The probability that a randomly chosen vector n is normal to a hyperplane H whose intersection with P does not pass between u and v (such as d in Figure 3) is precisely $\mathbf{angsim}(u, v) = \frac{\pi-\theta}{\pi}$. The intersection of H and P does not pass between u and v, iff. the dot products $n.u$ and $n.v$ of n with u and v have the same sign. It follows that, for a vector space $\mathcal{U}$, the family of functions $\mathcal{F} = \{f_w \mid w \in \mathcal{U}\}$ defined as follows is an **angsim**-sensitive family: for any u and w in $\mathcal{U}$, $f_w(u) = sign(w.u)$.

When we try to apply this classical version of random projection to instance-based type matching, we observe that the computation of the dot product requires a set of random vectors whose size is equal to the total number of distinct

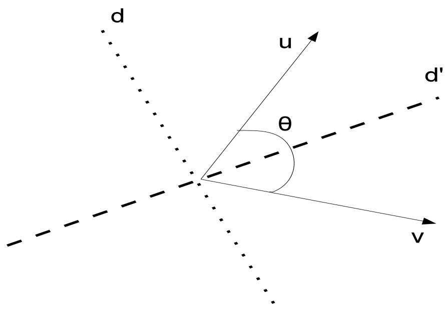


Fig. 3. Angular Similarity between two vectors

terms in all **termlist**(t) for $t \in \mathcal{T}$. However, this **angsim**-sensitive family is impractical for the following reasons:

– First, it is inefficient because it requires storing, in main memory, very large dimensional randomly generated vectors. For example, for a ±3.2% error margin at 95% confidence level, about 1000 such vectors need to be kept (or regularly brought) in main memory. For large dataset such as Linked Data, the number of distinct terms (*i.e.*, the dimension of the normal vectors) can be quite large. For reference, the number of words in the English language is estimated to be slightly above one million.
– Second, it requires that the total number of distinct terms (*i.e.*, the dimension of the randomly selected vectors) must be known in advance - before any signature computation or similarity estimation can be performed. This is a significant hurdle for an efficient distributed and streaming implementation, where the similarity between two types t and t' can be computed as soon as all the terms in **termlist**(t) and **termlist**(t') have been observed.

To address these two limitations, we consider a different **angsim**-sensitive family for our problem of instance-based type matching. Given a universal hash family $\mathcal{H}$ [12], a more efficient **angsim**-sensitive family $\mathcal{F}' = \{ f'_h \mid h \in \mathcal{H} \}$ to sample from in our case is defined as follows (with **termset**$(t) = \{r \mid r \in$ **termlist**$(t)\}$): for $t \in \mathcal{T}$ and $h \in \mathcal{H}$,

$$f'_h(t) = \begin{cases} +1 \text{ if } \sum_{r \in \mathbf{termset}(t)} \mathbf{termvec}(t)[r] \times h(r) \geq 0 \\ -1 \text{ otherwise} \end{cases}$$

Thus, randomly selecting elements of $\mathcal{F}'$ is equivalent to randomly selecting elements of $\mathcal{H}$. Note that $f'_h \in \mathcal{F}'$ is the same as $f_w \in \mathcal{F}$ where, for a term r, $w[r] = h(r)$ ($w[r]$ denotes the weight of the term r in vector w).

The **angsim**-sensitive family $\mathcal{F}'$ addresses the issues of the standard **angsim**-sensitive family $\mathcal{F}$ in two ways. First, instead of storing, in main-memory, very lengthy random vectors, we simply need to store n random hash functions $h \in \mathcal{H}$.

In practice, we use the set of 1000 Rabin Fingerprints [9] of degree 37 (*i.e.*, irreducible polynomials of degree 37) as $\mathcal{H}$. Each randomly selected element of $\mathcal{H}$ can thus be stored in less than 64 bits. Second, for two types t and t', their signatures can be computed in a streaming fashion, which means we can update the signature for each type as the instance values of that type are read incrementally. With the **angsim**-sensitive family of functions, signature construction can be done independently for each type and is therefore fully parallelizable. In practice, we implemented the random projection method with Hadoop.

3.2 Jaccard Similarity (MinHash)

For a type t, the list of tokenized terms for instances of the type $\mathbf{termlist}(t)$ may contain repeated terms. Taking into account term repetition enables us to distinguish between types with the same set of terms while the terms have different frequencies between the two types. For each type t in the finite set $\mathcal{T}$ of types, we associate a set of occurrence annotated terms $\mathbf{termOccSet}(t) = \{r : k \mid r \in \mathbf{termlist}(t)\ \&\ 1 \leq k \leq \mathbf{occ}(r, \mathbf{termlist}(t))\}$ where $\mathbf{occ}(r, l)$ denotes the number of occurrences of a term r in the list l. An occurrence annotated term $r : k$ of a type t corresponds to the k^{th} occurrence of the term r in the list $\mathbf{termlist}(t)$ of terms of t. We then measure the similarity of two types using Jaccard similarity on their instance values. Traditionally, the Jaccard similarity $\mathbf{jaccsim}(u, v)$ of two sets u and v elements of $\mathcal{U}$ is defined as the ratio of the size of the intersection of the two sets divided by the size of their union: $\mathbf{jaccsim}(u, v) = \frac{|\ u\ \cap\ v\ |}{|\ u\ \cup\ v\ |}$. To address the scalability issues, we employ MinHash, a standard LSH technique to estimate the Jaccard similarity over the set $\mathcal{U}$ of all the sets of occurrence annotated terms $\mathbf{termOccSet}(t)$ with $t \in \mathcal{T}$.

MinHash considers a set $\mathcal{F}$ of hash functions where each function h^{min} maps from $\mathcal{U}$ to the set $\mathbb{Z}$ of integers as follows: for $u \in \mathcal{U}$,$h^{min}(u) = min_{x \in u} h(x)$, where h in a hash function from a universal hash family $\mathcal{H}$[1] from $\mathcal{U}$ to $\mathbb{Z}$. $h^{min}(u)$ computes the minimal value of h on elements of u. Now, given two elements u and v of $\mathcal{U}$, $h^{min}(u) = h^{min}(v)$ iff. the minimal value of h in the union $u \cup v$ is also contained in the intersection $u \cap v$. It follows that the probability that the MinHash values for two sets are equal is equivalent to their Jaccard similarity: $\mathbf{Pr}(h^{min}(u) = h^{min}(v)) = \mathbf{jaccsim}(u, v)$. Thus, $\mathcal{F} = \{h^{min} \mid h \in \mathcal{H}\}$ is a **jaccsim**-sensitive family of functions. Then the Jaccard similarity of two sets can be estimated with the percentage of n such functions whose MinHash values are equal. Note that the transformation and MinHash computation for each type is independent of other types, so they can be parallelized in a distributed computing framework like Hadoop.

In addition to computing the Jaccard similarity between two types t and t', we observe that it is important to measure their containment, particularly when the sizes of $\mathbf{termOccSet}(t)$ and $\mathbf{termOccSet}(t')$ are very different. For two types t and t', $C_{t \subseteq t'} = \frac{|t \cap t'|}{|t|}$ measures the containment of t in t'. It is equal to

[1] Elements of $\mathcal{H}$ are assumed to be *min-wise independent*: for any subset of $\mathcal{U}$, any element is equally likely to be the minimum of a randomly selected function $h \in \mathcal{H}$.

1 iff. t is contained in t'. It can be expressed in terms of the Jaccard similarity as follows:

$$C_{t \subseteq t'} = \frac{\textbf{jaccsim}(t,t')}{\textbf{jaccsim}(t,t')+1} \times (1 + \frac{|t'|}{|t|})$$

3.3 Banding Technique to Avoid Pairwise Comparison

Recall that to apply the banding technique, conceptually we construct a signature matrix for all types, with each column representing a type and the rows computed from n independent hash functions. Through the banding technique, we can generate candidate similar pairs; a pair of types becomes candidate for further computation when they agree in at least one band. For each candidate pair, we could store both the type URIs and the associated signatures, and distribute the actual similarity computation based on signatures across a cluster of machines. However, it raises a strong requirement for both disk I/Os and network I/Os in a distributed setting; note that the number of candidate pairs could still be huge using the LSH technique and each pair of signatures take nonignorable space. The way we address the challenge is to split the similarity computation in two phases. In phase one, we generate candidate pairs in the format of (`type-uri1`, `type-uri2`). A join of candidate type pairs with the signature matrix will produce a new type pairs in the format of (`type-uri1+signature1`, `type-uri2`). In phase two, another join of the newly generated type pairs with the signature matrix will do the actual computation of similarity based on the signatures associated with `type-uri1` and `type-uri2`.

4 Evaluation

In this section, we report the results of applying LSH techniques to find related pairs of types in Freebase[2] (retrieved on May 2012) and DBpedia [1] (version 3.6). Freebase data contains 22,091,640 entities (or topics), and DBpedia contains 1,668,503 entities (or things). These entities have overall 44,389,571 label values in Freebase and 2,113,596 labels in DBpedia. Since our goal is matching types based on instances, we prune those types that have less than k number of instances. For the results reported in this section, we have $k = 500$ which reduces the number of types in Freebase from 15,364 to 1,069, and from 238 to 155 in DBpedia. We further restrict the matching to label properties of the instances (`rdfs:label` in DBpedia and /object/name in Freebase). The resulting sets of types have overall 43,919,815 values in Freebase and 2,042,337 values in DBpedia, which means on average 37,551 values per type. Notice that when compared to the values before the pruning, the pruned datasets retain 98% and 96% of their values, respectively. The actual number of instance values for each type could vary significantly for each type. For example, there are 1,847,416 persons on Freebase, and 363,752 persons in DBpedia.

[2] `http://www.freebase.com/`

For the reported results, we merge the DBpedia and Freebase datasets and match all the types in both with themselves. This allows us to evaluate the effectiveness of the LSH techniques in discovering related types within Freebase, within DBpedia, and between Freebase and DBpedia. For purposes of the evaluation, we eliminate type pairs that match a type with itself from our analysis. We fix the number of hash functions to 500 for MinHash and 1,000 for RHP.

4.1 Discovering Equivalence and Containment Relations

We first measure the effectiveness of our approach in discovering two kinds of relationships between types: *equivalence* (i.e., two types refer to similar real-world concept) and *containment* (i.e., one type is a subclass of the other). Unfortunately, there are no manual type matchings between DBpedia and Freebase, although there are instance matches that are connected with `owl:sameAs` links. We therefore need to derive ground truth for matching Freebase with DBpedia types, using the existing `owl:sameAs` links at the instance level between the two data sources. We include a pair of types in ground truth if and only if their sets of instances are linked with at least a given number, θ_g , of `owl:sameAs` links, to ensure we include valid type pairs in the ground truth. We call θ_g the ground truth cardinality threshold. The ground truth for discovery of equivalent and containment types within a single source is derived similarly by finding the number of shared instances between types. A pair of types is included in the ground if an only if there are at least θ_g number of instances that exist in both types (e.g., if θ_g number of instances have both `dbpedia:Person` and `dbpedia:Actor` as their types, the type pair will be included in the ground truth).

We use the traditional information retrieval accuracy measures, namely precision, recall and F-measure. Precision is the ratio of correct results to all results retrieved. Recall is the percentage of results in the ground truth that are actually retrieved. The F-measure is defined as the harmonic mean of precision and recall, calculated as $F = \dfrac{2 \times Precision \times Recall}{Precision + Recall}$. For the type matches based on Jaccard or cosine similarity, we need a similarity threshold to define non-matches. However, there is no fixed threshold value that works best across all the types, which makes threshold selection ad-hoc. A common approach in deciding matches is to sort the matching results in a descending order of the similarity score, and pick only the top-k results. Again, the value of k can be different for each type. For this evaluation, we set the value of k for each type t as the number of types t' that match with t in the ground truth; in our ground truth, this value varies from 1 to 86, with an average of 4. We call the resulting measures variable-k top-k precision, recall and F-measure.

Table 1 shows the variable-k top-k precision, recall and F-measure obtained with $\theta_g = 1,000$. For all cases, RHP outperforms MinHash in terms of accuracy. Note that these results are obtained without any post-processing, but by sorting the results based on the estimated Jaccard/cosine similarity values respectively. The superiority of the cosine metric over Jaccard suggests that *tf*idf* is an effective term weighting strategy for instance-based type matching.

However, a key advantage for Jaccard is that it gives us an indication of whether there is a containment relationship between two types, which cannot be derived from cosine similarity. As we discussed earlier, when the sets of instance values for two types are very different in size, the maximal similarity computed by either Jaccard will be significantly below 1, even if one of the sets is perfectly contained in the other. To discover containment relationship between types, we add a post-processing phase to MinHash that sorts the output type pairs by an estimation of containment ratio $C_{u \subseteq v}$ from one type to the other, as discussed in Section 3.2. A key problem in measuring accuracy is again the lack of ground truth. We derived the ground truth for both Freebase and DBpedia using the following method. We include into the ground truth a pair of types t_1 and t_2 if the ratio of the number of instances that have both t_1 and t_2 as their types to the number of instances of type t_2 is above a threshold θ_c. For the results reported in this section, we set $\theta_c = 0.8$, which means that we have (t_1, t_2) in the ground truth if 80% of instances of type t_1 also have t_2 as their types. For DBpedia, in addition to the ground truth derived similarly, we use the set of all subclass relationships in the type hierarchy in the DBpedia ontology as our ground truth. Note that using the DBpedia ontology as a ground truth is a very conservative approach. In DBpedia, there is a strict type hierarchy such that *Song* is a *Musical Work* which is in turn a *Work*, and *Work* is in fact a *Thing*. None of the actual instances of Song are annotated with all their superclasses (e.g. *Thing*). But our approach on instance-based type matching requires that instances be annotated with both subclasses and superclasses in order to find containment. Therefore using the DBpedia ontology is a very conservative ground truth, but we nevertheless include it for evaluation purposes. Table 2 shows the accuracy results for the three cases. The results for DBpedia with the derived ground truth are far better than those for DBpedia with the ontology as the ground truth. The results for Freebase are overall worse than that for DBpedia reflecting a trend we saw in Table 1. We discuss possible reasons for this later.

We also measured the effectiveness of the LSH technique in pruning out a large number of irrelevant type pairs (i.e., those with low similarity values) from analysis. To quantify this pruning effect, we define a measure called *Reduction Ratio (RR)*. This measure is calculated as the ratio of the number of type pairs for which similarity was computed by the LSH techniques, to the total number of possible type pairs. $(1 - RR)$ indicates the amount of computation that is saved by the use of LSH and captures the degree to which LSH was effective for actual type matching performance. Recall that LSH depends on (a) a similarity threshold (type pairs with similarity values below it are considered irrelevant) and (b) the user's tolerance for a false negative rate (*i.e.*, the degree to which recall is important to the user). Our experiments were run very conservatively, with very low similarity thresholds (1% for MinHash and 10% for RHP), and false negative rates of 10% for MinHash and 5% for RHP. Yet, we obtained an overall 77% savings in computation (*i.e.*, $RR = 23\%$): only 354,404 type comparisons were performed out of the 1,498,176 total possible pairs of types. Table 1 shows the reduction ratio, RR at Max, achieved at the maximum possible

Table 1. Accuracy of Discovering Equivalent/Subclass Types in Freebase and DBpedia

	Freebase-DBpedia		Freebase-Freebase		DBpedia-DBpedia	
	MHash	RHP	MHash	RHP	MHash	RHP
Top-1 Precision	73.7%	75.0%	54.7%	59.9%	84.6%	100.0%
Top-k Precision	48.9%	61.7%	53.8%	60.7%	86.7%	100.0%
Top-k Recall	69.0%	80.4%	17.3%	19.9%	76.5%	82.4%
Top-k F-score	57.2%	69.8%	26.2%	30.0%	81.3%	90.3%
Overall Recall	84.8%	90.5%	87.7%	93.5%	94.1%	94.1%
RR at Max	0.21	0.18	0.22	0.13	0.34	0.05
RR at 80%	0.15	0.04	0.19	0.03	0.05	0.05

value of recall (*i.e.*, for our very conservative setup with little tolerance for false negatives), and, RR obtained if the threshold and the acceptable false negative rate were adjusted to produce about 80% recall value. For example, for Freebase-Freebase case, in our conservative setup where we were willing to accept about 5% false negative rate for RHP, we achieved a 93.5% recall with RHP. At this recall, reduction ratio is 0.13, *i.e.*, similarity estimation was actually computed for 13% of the total number of type pairs. However, if 80% recall is acceptable (*i.e.*, a higher acceptable false negative rate), reduction ratio of 0.03 (i.e., 97% in savings) can be achieved.

In addition, we compared the running time of similarity computation using MinHash and RHP signatures, with the exact Jaccard and cosine similarity computation, using a subset of 200 types (100 types from each dataset) containing overall 3,690,461 label values (average 18,452 values per type). We picked this smaller subset to make it feasible to run the exact computation on a single machine for a fair comparison with an implementation of the LSH techniques that does not take advantage of our Hadoop-based implementation. The experiment was run on a Linux machine with 4 cores and 24GB of memory. The exact Jaccard similarity computation took 1,666.28 seconds (28 minutes) while the similarity computation using the MinHash signatures took only 0.409 seconds. The error in similarity scores was 0.008, and overall 4,958 similarity computations (out of the possible 40,000) were performed. The exact computation returned 1,152 pairs, and MinHash results missed only 30 of these pairs which means a recall of 97.4%. The exact cosine similarity took 302.06 seconds to run while the RHP-based computation took 2.41 seconds. The average error in cosine similarity scores was 0.025, but given our conservative settings which results in block size 2, no calculations were saved as a result of banding, and therefore the recall was 100%.

We next turn to investigate the differences in the evaluation results for inferring equivalence/subclass relations. As shown in Table 1, the similarity measures we used achieve good accuracy in the Freebase-DBpedia and DBpedia-DBpedia cases, but not in the Freebase-Freebase case. As we will explain in the next section, the reason for this low accuracy is not poor performance of the similarity measure, but existence of several *related types* that share similar instance values and therefore are returned as false positives (e.g., *Actor* and *Music Director*).

Table 2. Accuracy of Discovering Containment Relationship in Freebase and DBpedia

	DBpedia-Derived	DBpedia-Ontology	Freebase-Derived
Top-1 Precision	85.7%	81.3%	74.4%
Top-k Precision	87.0%	85.0%	63.3%
Top-k Recall	55.1%	24.4%	22.0%
Top-k F-score	67.4%	37.9%	32.6%
Overall Recall	55.1%	24.4%	22.6%

4.2 Discovering Semantically Related Types

To investigate the reason behind lower precision in Freebase-Freebase case, we manually inspected a subset of the results. Upon manual inspection, we observed that a large portion of wrong matches in top-k results are between types that are not equivalent, but are semantically related. For example, the two types represent person entities (e.g., type `athlete` linked to type `tv_actor`), or in general, the two types are a subclass of a single type. Based on this observation, we extended the ground truth for Freebase-Freebase case to include such type pairs. We first derive subclass relations from instance data, and then add the pairs of types that are subclasses of the same type to the ground truth. This improved the variable-k top-k precision score to 66.4% in MinHash with overall recall at 90.0%, and 69.0% in RHP with overall 78.2% recall. In this case, we observe that MinHash performs almost as well as RHP in terms of precision, with better recall. This shows that Jaccard similarity and therefore MinHash are more suitable in discovering semantically related types.

In our manual inspection of the results, we asked four members of our team to individually evaluate a set of 144 pairs of types in the Freebase-Freebase results, that are the top 5 (or less) matches for 50 random types in Freebase. One of the four members performed the evaluation solely by looking at type names, while others also inspected a sample of instance values. The evaluator based only on type names found only 33 out of the 144 pairs accurate, while others found between 77 and 92 type pairs accurate. The difficulty of matching by type names only in these scenarios arises both from types with only machine generated identifiers (*e.g.*, `http://rdf.freebase.com/rdf/m/06vwzp1`) and from types with obscure human generated identifiers (*e.g.*, the instances `http://rdf.freebase.com/rdf/base/database2/topic` are famous mountains). For the evaluations based on inspection of instance values, 2 out of 3 evaluators agreed on 88 of the type pairs, for a precision of 61.1%.

5 Related Work

The problem of matching database schema and ontologies with the goal of finding elements that are semantically related has been studied extensively in the past. In the existing categorization of schema and ontology matching techniques [22,31,33], our approach falls into the purely *instance-based* and *element-level* category, as we rely on instance values rather than element labels and schema structure and information. Our proposed techniques improve the scalability of a technique referred to as *similarity-based disjoint extension comparison* [22], which unlike the *common extension comparison* technique [26,28] does not require the classes to share the same set of instances. Our technique is unsupervised and can be used to extend existing rule-based matching systems. The inability to effectively exploit data instances has been recognized as the main drawback of rule-based techniques [17, page 86]. In the categorization provided by Kang and Naughton [27], our method falls into the interpreted class of matching techniques since we rely on an interpretation of instance values. However, unlike the majority of interpreted matchers, we do not rely on attribute names, nor do we rely on learning and the availability of training data.

Examples of matching systems that use instance values in matching are COMA++ [2,21], SEMINT [29], LSD [16], Autoplex [4], Glue [18], and DUMAS [6]. Of these, only COMA++ and DUMAS are unsupervised. COMA++ supports two instance-based matchers in addition to several schema-based methods: 1) constraint-based matching methods that consider characteristics or patterns in instance values such as type, average length, and URL or email patterns; 2) a content-based matcher that builds a similarity matrix by performing a pair-wise comparison of all instance values and aggregating the result. Our approach can be seen as a way of making such a content-based matcher scalable. To the best of our knowledge, we are the first to address the scalability of such *all-to-all* instance-based matching. DUMAS relies on identification of duplicate records by string similarity matching in advance to improve the accuracy and efficiency of the approach. The attribute identification framework proposed by Chua et al [25] uses duplicates that are identified by matching key identifier attributes (as opposed to string matching) and takes into account several properties of attributes derived from instance values.

There are also instance-based approaches that do not rely on overlapping or similar instance values, but take into account the correlation between attributes or their properties. Notably, Kang and Naughton [27] propose a two-step matching that first builds a dependency graph for the attributes in each data source by mining the instance values, and then uses a graph matching algorithm to find attribute correspondences. Dai et al [14] propose an information-theoretic measure to validate the matching results that can work even if both schema information and instance values do not match (e.g., merging of two customer databases from companies that do not share any customers). In our work, our goal is to match elements (types) only if their instance values are *similar*. The approach presented in this paper can be used as a part of large-scale data integration and analytics systems [23] and link discovery systems [24,8]. Our work is motivated by the massive growth in the amount of data available on the web, and our experience in matching large enterprise repositories [11,19,20].

For an overview of other schema-based and instance-based techniques, refer to existing survey articles and books [3,22,27,31,33].

6 Conclusion

We present an instance-based type matching approach based on locality-sensitive hashing and evaluate it on linking two large Linked Data sources on the web. We show how LSH techniques are very effective in pruning large numbers of irrelevant type comparisons, and point to how they can be deployed for type matching and type containment.

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., 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.) ISWC/ASWC 2007. LNCS, vol. 4825, pp. 722–735. Springer, Heidelberg (2007)
2. Aumueller, D., Do, H.H., Massmann, S., Rahm, E.: Schema and Ontology Matching with COMA++. In: ACM SIGMOD Int'l Conf. on Mgmt. of Data, pp. 906–908 (2005), System demonstration
3. Bellahsene, Z., Bonifati, A., Rahm, E.: Schema Matching and Mapping (Data-Centric Systems and Applications), 1st edn. Springer (2011)
4. Berlin, J., Motro, A.: Database Schema Matching Using Machine Learning with Feature Selection. In: Pidduck, A.B., Mylopoulos, J., Woo, C.C., Ozsu, M.T. (eds.) CAiSE 2002. LNCS, vol. 2348, pp. 452–466. Springer, Heidelberg (2002)
5. Bernstein, P.A., Melnik, S., Petropoulos, M., Quix, C.: Industrial-Strength Schema Matching. SIGMOD Record 33(4), 38–43 (2004)
6. Bilke, A., Naumann, F.: Schema Matching Using Duplicates. In: IEEE Proc. of the Int'l Conf. on Data Eng., pp. 69–80 (2005)
7. Bizer, C., Jentzsch, A., Cyganiak, R.: State of the LOD Cloud (September 2011), `http://www4.wiwiss.fu-berlin.de/lodcloud/state/` (online; accessed October 31, 2011)
8. Bizer, C., Volz, J., Kobilarov, G., Gaedke, M.: Silk - A Link Discovery Framework for the Web of Data. In: WWW 2009 Workshop on Linked Data on the Web (LDOW 2011) (April 2009)
9. Broder, A.Z.: Some applications of rabin's fingerprinting method. In: Sequences II: Methods in Communications, Security, and Computer Science (MCSCS), pp. 143–152. Springer (1993)
10. Broder, A.: On the resemblance and containment of documents. In: Proc. Compression and Complexity of Sequences, pp. 21–29 (1997)
11. Byrne, B., Fokoue, A., Kalyanpur, A., Srinivas, K., Wang, M.: Scalable matching of industry models - a case study. In: Proceedings of the International Workshop on Ontology Matching, OM (2009)
12. Carter, J., Wegman, M.N.: Universal classes of hash functions. Journal of Computer and System Sciences 18(2), 143–154 (1979), `http://www.sciencedirect.com/science/article/pii/0022000079900448`
13. Charikar, M.: Similarity estimation techniques from rounding algorithms. In: ACM Symp. on Theory of Computing (STOC), pp. 380–388 (2002)
14. Dai, B.T., Koudas, N., Srivastava, D., Tung, A.K.H., Venkatasubramanian, S.: Validating Multi-column Schema Matchings by Type. In: IEEE Proc. of the Int'l Conf. on Data Eng., pp. 120–129 (2008)
15. Do, H.H., Rahm, E.: COMA - A System for Flexible Combination of Schema Matching Approaches. In: Proc. of the Int'l Conf. on Very Large Data Bases (VLDB), pp. 610–621 (2002)

16. Doan, A., Domingos, P., Halevy, A.Y.: Reconciling Schemas of Disparate Data Sources: A Machine-Learning Approach. In: ACM SIGMOD Int'l Conf. on Mgmt. of Data, pp. 509–520 (2001)
17. Doan, A., Halevy, A.Y.: Semantic Integration Research in the Database Community: A Brief Survey. AI Magazine 26(1), 83–94 (2005)
18. Doan, A., Madhavan, J., Domingos, P., Halevy, A.Y.: Ontology Matching: A Machine Learning Approach. In: Handbook on Ontologies, pp. 385–404. Springer (2004)
19. Duan, S., Fokoue, A., Srinivas, K.: One Size Does Not Fit All: Customizing Ontology Alignment Using User Feedback. 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. 177–192. Springer, Heidelberg (2010)
20. Duan, S., Fokoue, A., Srinivas, K., Byrne, B.: A Clustering-Based Approach to Ontology Alignment. 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. 146–161. Springer, Heidelberg (2011)
21. Engmann, D., Maßmann, S.: Instance Matching with COMA++. In: BTW Workshops, pp. 28–37 (2007)
22. Euzenat, J., Shvaiko, P.: Ontology Matching. Springer (2007), http://book.ontologymatching.org/
23. Hassanzadeh, O., Duan, S., Fokoue, A., Kementsietsidis, A., Srinivas, K., Ward, M.J.: Helix: Online Enterprise Data Analytics. In: Proceedings of the 20th International World Wide Web Conference (WWW 2011) - Demo Track (2011)
24. Hassanzadeh, O., Xin, R., Miller, R.J., Kementsietsidis, A., Lim, L., Wang, M.: Linkage Query Writer. Proceedings of the VLDB Endowment (PVLDB) 2(2), 1590–1593 (2009)
25. Huang, C.C.E., Chiang, R.H.L., Lim, E.P.: Instance-based attribute identification in database integration. VLDB J. 12(3), 228–243 (2003)
26. Isaac, A., van der Meij, L., Schlobach, S., Wang, S.: An Empirical Study of Instance-Based Ontology Matching. 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.) ISWC/ASWC 2007. LNCS, vol. 4825, pp. 253–266. Springer, Heidelberg (2007)
27. Kang, J., Naughton, J.F.: On Schema Matching with Opaque Column Names and Data Values. In: ACM SIGMOD Int'l Conf. on Mgmt. of Data, pp. 205–216 (2003)
28. Kirsten, T., Thor, A., Rahm, E.: Instance-Based Matching of Large Life Science Ontologies. In: Cohen-Boulakia, S., Tannen, V. (eds.) DILS 2007. LNCS (LNBI), vol. 4544, pp. 172–187. Springer, Heidelberg (2007)
29. Li, W.S., Clifton, C.: SEMINT: A tool for identifying attribute correspondences in heterogeneous databases using neural networks. Data and Knowledge Engineering 33(1), 49–84 (2000)
30. Madhavan, J., Bernstein, P.A., Rahm, E.: Generic Schema Matching with Cupid. In: Proc. of the Int'l Conf. on Very Large Data Bases (VLDB), pp. 49–58 (2001)
31. Rahm, E., Bernstein, P.A.: A Survey of Approaches to Automatic Schema Matching. The Int'l Journal on Very Large Data Bases 10(4), 334–350 (2001)
32. Rajaraman, A., Ullman, J.D.: Mining of Massive Datasets, 1st edn. Cambridge University Press, College Station (2011)
33. Shvaiko, P., Euzenat, J.: A Survey of Schema-Based Matching Approaches. In: Spaccapietra, S. (ed.) Journal on Data Semantics IV. LNCS, vol. 3730, pp. 146–171. Springer, Heidelberg (2005)

Automatic Typing of DBpedia Entities

Aldo Gangemi[1], Andrea Giovanni Nuzzolese[1,2], Valentina Presutti[1], Francesco Draicchio[1], Alberto Musetti[1], and Paolo Ciancarini[1,2]

[1] STLab-ISTC Consiglio Nazionale delle Ricerche, Rome, Italy
[2] Dipartimento di Scienze dell'Informazione, Università di Bologna, Italy

Abstract. We present *Tìpalo*, an algorithm and tool for automatically typing DBpedia entities. Tìpalo identifies the most appropriate types for an entity by interpreting its natural language definition, which is extracted from its corresponding Wikipedia page abstract. Types are identified by means of a set of heuristics based on graph patterns, disambiguated to WordNet, and aligned to two top-level ontologies: WordNet supersenses and a subset of DOLCE+DnS Ultra Lite classes. The algorithm has been tuned against a golden standard that has been built online by a group of selected users, and further evaluated in a user study.

1 Introduction

Wikipedia is a large-scale resource of content capturing encyclopedic knowledge collaboratively described by the crowds. Entities described in Wikipedia are formally represented in DBpedia, the RDF translation of information from many localized versions of Wikipedia, eminently the English one. There are DBpedia datasets providing types for entities, but a large number of them is still untyped, or has a very specialized type, and types are taken from ontologies that have heterogeneous granularities or assumptions (e.g., 272 infobox-based types in the DBpedia ontology (DBPO)[1] against almost 290,000 category-based in YAGO [17]). This situation makes it difficult to identify a proper reference ontology for Wikipedia, or to reuse DBpedia knowledge with a good precision. While it is reasonable to have limited semantic homogeneity on the Web, it is highly desirable to bring a more organized and complete typing to DBpedia entities. Knowing *what* a certain entity is (e.g., a person, organization, place, instrument, etc.) is key for enabling a number of desirable functionalities such as *type coercion* [10], data pattern extraction from links [14], entity summarization (cf. Google Knowledge Graph), automatic linking, etc.

The two *de facto* reference ontologies for DBpedia resources are currently DBPO and YAGO. Both provide types for DBpedia entities, and in particular YAGO has high performances as far as typing quality is concerned. However, their coverage is partial, both extensionally (number of typed resources), and intensionally (conceptual completeness), since they rely on Wikipedia categories, and infoboxes (that are not included in all Wikipedia pages). In addition, the

[1] `http://dbpedia.org/ontology/`

number of resources that could be typed from Wikipedia content is even larger than the number of Wikipedia pages: for example, many DBpedia entities are referenced by fragments of Wikipedia pages. Our aim is to enable automatic typing of entities by exploiting the natural language (NL) definitions from their corresponding Wikipedia pages. Hence, without relying on categorization, or on the presence of structured data such as Wikipedia infoboxes.

Although there are numerous Natural Language Processing (NLP) approaches to learning ontologies from text, they need training phases that can take a long time, and may need a huge manually annotated corpus in order to perform training. When dealing with large-scale corpora such as Wikipedia, we need to identify sustainable procedures as far as speed is concerned. Therefore, none of the existing NLP resources (cf. Section 2) can be directly used to perform the automatic typing of DBpedia entities that we propose here.

We present *Tìpalo*, a tool that automatically assigns types to DBpedia entities based on their NL definitions as provided by their corresponding Wikipedia pages. We use a tool (cf. FRED [16]) that implements deep parsing methods based on frame semantics for deriving RDF and OWL representations of NL sentences. On top of it, we have implemented a procedure to extract types from the RDF representation of definitions. The procedure is tailored to Wikipedia pages, and reuses a tool for word sense disambiguation (cf. UKB [1]) to automatically link the extracted types to WordNet. We also use alignments of WordNet to two top-level ontologies: WordNet *super senses*, and DUL+DnS Ultralite[2].

Results show that Tìpalo can extend typing of DBpedia entities with high accuracy, and support the incremental definition of a Wikipedia ontology that emerges from what is written in the articles, rather than from metadata or statistical observations.

The contribution of this paper can be summarized as follows:

– a tool, named *Tìpalo*, implementing a process for automatically typing DBpedia entities, based on their NL definitions, which is fast enough to be used in realistic projects, while performing with good precision and recall;
– a sample Wikipedia ontology, incrementally built with Tìpalo, encoded in two semantic resources: (i) the Wikipedia entity types dataset, containing automatically typed (and evaluated) DBpedia entities extracted from 627 definitions; (ii) the Wikipedia class taxonomy dataset, including WordNet types, WordNet super senses, DUL types, and new defined types put in a `rdfs:subClassOf` taxonomy;
– an updated mapping between WordNet 3.0 synsets and top-level ontology classes, released as RDF datasets;
– a golden standard of 100 typed entities, manually annotated through a collaborative effort supported by an online tool that manages user agreement.

The whole procedure can be executed by means of a web service[3]. In order to favor reuse and repeatability of our approach and experiments, the web service,

[2] http://www.ontologydesignpatterns.org/ont/dul/DUL.owl
[3] http://wit.istc.cnr.it/stlab-tools/tipalo

tools, and resources that we have produced are all publicly available from the Wikipedia Ontology page[4].

2 Related Work

The main approaches for typing DBpedia entities are: (i) the DBpedia project [11], which manually created a DBpedia ontology (DBPO) based on a limited number of Wikipedia infobox templates. Currently, DBpedia entities having DBPO types are ∼1.83M[5] (version 3.7), against almost 4M Wikipedia pages (August 2012). Besides its limited extensional coverage, DBPO suffers from limited intensional coverage [15] due to the manual extraction procedure on infobox templates that exist only for a subset of Wikipedia page types; (ii) YAGO [17], an ontology extracted from Wikipedia categories and infoboxes, and aligned to a subset of WordNet. YAGO's coverage is larger than DBPO (∼2.7M entities), however still incomplete and its intensional completeness is affected by its reliance on Wikipedia infoboxes and categories. In [15] learning techniques as well as rule-based approaches for automatic typing of DBpedia entities have been analyzed. The analysis confirmed the difficulty of this task, and highlighted the limits posed by the reliance on Wikipedia categories and infoboxes.

Relevant work related to our method includes Ontology Learning and Population (OL&P) techniques [2]. Typically OL&P is implemented on top of machine learning methods, hence it requires large corpora, sometimes manually annotated, in order to induce a set of probabilistic rules. Such rules are defined through a training phase that can take a long time. Examples of such methods include [3,20,18]. All these methods would be hardly applicable to large corpora such as Wikipedia due to the time and resources they require, if all the potential types present in NL descriptions need to be extracted. Other approaches to OL&P use either lexico-syntactic patterns [9], or hybrid lexical-logical techniques [19], but to our knowledge no practical tools have emerged so far for doing it automatically while preserving high quality of results. [5] works great for large-scale information extraction centered on binary relation extraction. However, its resulting triplet graphs are not interlinked and feature a low recall of relevant syntactic structures, making it too limited for the automatic typing task.

The method presented in this paper differs from most existing approaches, by relying on a component named FRED [16], which implements a logical interpretation of NL represented in Discourse Representation Theory (DRT). FRED is fast and produces an OWL-based graph representation of an entity description, including a taxonomy of types. We parse FRED's output graph, and apply a set of heuristics, so that we can assign a set of types to an entity in a very efficient way. FRED is an example of *machine reading*.

Terms used for describing entity types are often polysemous i.e. they can have more than one meaning. We have empirically observed (on a sample of ∼800 entity definitions) that polysemous terms occur in 70% of descriptions; hence,

[4] The Wikipedia ontology page `http://www.stlab.istc.cnr.it/WikipediaOntology/`

[5] M → millions.

the word sense disambiguation (WSD) task is relevant in this context. A good survey of WSD methods is [12]. We use UKB [1] that shows high accuracy, with some impact on the speed of the process (cf. Section 5). A promising resource for disambiguation is BabelNet [13], which has produced a substantial alignment between WordNet and Wikipedia concept-like entities. However, its currently available dataset is not suitable for implementing a direct WSD functionality.

As a final remark, we mention a recent work on terminology extraction [8] which describes a number of wikipedia markup conventions that are useful hooks for defining heuristics. Some of them have been reused in our tool.

3 Data Sources

In the context of this work, we have used and produced a number of resources.

Wikipedia and DBpedia. Wikipedia is a collaboratively built multilingual encyclopedia on the Web. Each Wikipedia page usually refers to a single entity, and is manually associated to a number of categories. Entities referenced by Wikipedia pages are represented in DBpedia, the RDF version of Wikipedia. Currently, English Wikipedia contains 4M articles[6], while DBpedia wikilink dataset counts ∼15M distinct entities (as of version 3.6). One main motivation of this big difference in size is that many linked resources are referenced by *sections* of Wikipedia pages, hence lacking explicit categorization or infoboxes. However they have a URI, and a NL description, hence they are a rich source for linked data. Out of these ∼15M resources, ∼2.7 are typed with YAGO classes and ∼1.83M are typed with DBpedia classes. We use Wikipedia page contents as input for the *definition extractor* component (cf. Section 4), for extracting entity definitions.

WordNet 3.0 and WordNet 3.0 supersense RDF. WordNet[7] is a large database of English words. It groups words into sets of synonyms, called *synsets*, each expressing a different concept. Although WordNet includes different types of words such as verbs and adjectives, for the sake of this work we limit the scope to nouns. Words that may express different meanings, i.e. polysemous words, are related to different synsets. In this work, we use the WordNet 3.0 RDF porting[8] in order to identify the type of an entity. Hence when such type is expressed by a polysemous word we need to identify the most appropriate one. To this aim we exploit a WSD engine named UKB, as described in Section 4. Furthermore, WordNet 3.0 includes relations between synsets and supersenses, which are broad semantic categories. WordNet contains 41 supersenses, 25 of which are for nouns. We have produced a resource named *WordNet 3.0 Supersense RDF*[9] that encodes such alignments as RDF data. This RDF dataset is used by the *type matcher* (cf. Section 4) for producing triples relating entities and supersenses.

[6] Source: `http://en.wikipedia.org/wiki/Wikipedia:Size_of_Wikipedia`, Aug 2012.
[7] WordNet, `http://wordnet.princeton.edu/`
[8] `http://semanticweb.cs.vu.nl/lod/wn30/`
[9] `http://www.ontologydesignpatterns.org/wn/wn30/wordnet-supersense.rdf`

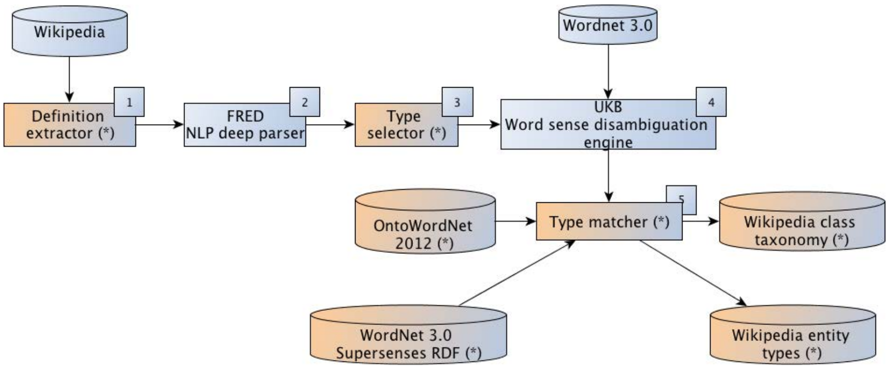


Fig. 1. Pipeline implemented by Tìpalo for automatic typing of DBpedia entities based on their natural language descriptions as provided in their corresponding Wikipedia pages. Numbers indicate the order of execution of a component in the pipeline. The output of a component i is passed as input to the next $i+1$ component. (*) denotes datasets and tools developed in this work, which are part of our contribution (cf. Section 3).

OntoWordNet (OWN) 2012 is a RDF resource that updates and extends OWN [7]. OWN is an OWL version of WordNet, which includes semantic alignments between synsets and DULplus types. DULplus[10], extends DUL[11], which is the OWL light version of DOLCE + DnS [6] foundational ontology. OWN 2012 contains mappings between 859 general synsets and 60 DULplus classes. Such mappings have been propagated through the transitive closure of the hyponym relation in order to cover all ∼82,000 synsets. In the context of this work, we have updated OWN to the WordNet 3.0 version, and performed a revision of the manual mapping relations. Furthermore, we have defined a lightweight foundational ontology called Dolce Zero[12], whose classes generalize a number of DULplus classes used in OWN. We have used a combination of 23 Dolce Zero and DULplus classes for building a sample Wikipedia ontology. The reduction to 23 classes has been made in order make it comparable to the WordNet supersense set, and to simplify the task of evaluators.

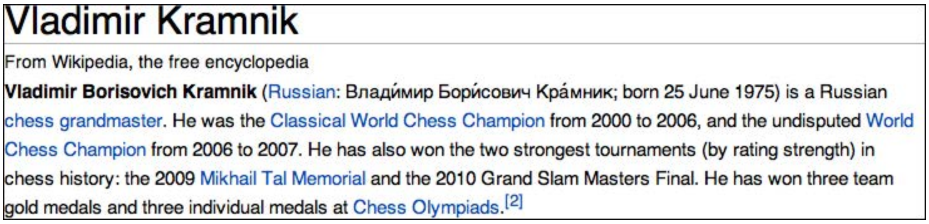
4 The Automatic Typing Procedure

Tìpalo is based on a pipeline of components and data sources, described below, which are applied in the sequence illustrated in Figure 1.

[10] Dolce Ultra Lite Plus ontology, `http://www.ontologydesignpatterns.org/ont/wn/dulplus.owl`

[11] Dolce Ultra Lite ontology, `http://www.ontologydesignpatterns.org/ont/dul/DUL.owl`

[12] Dolce Zero ontology, `http://www.ontologydesignpatterns.org/d0.owl`



Vladimir Kramnik

From Wikipedia, the free encyclopedia

Vladimir Borisovich Kramnik (Russian: Владимир Борисович Крáмник; born 25 June 1975) is a Russian chess grandmaster. He was the Classical World Chess Champion from 2000 to 2006, and the undisputed World Chess Champion from 2006 to 2007. He has also won the two strongest tournaments (by rating strength) in chess history: the 2009 Mikhail Tal Memorial and the 2010 Grand Slam Masters Final. He has won three team gold medals and three individual medals at Chess Olympiads.[2]

Fig. 2. First paragraph of the Wikipedia page abstract for the entity "Vladimir Kramnik"

1. Extracting definitions from Wikipedia pages (definition extractor). The first step, performed by the **definition extractor**, consists in extracting the definition of a DBpedia entity from its corresponding Wikipedia page abstract. We identify the shortest text including information about the entity type. Typically, an entity is defined in the first sentence of a Wikipedia page abstract, but sometimes the definition is expressed in one of the following sentences, can be a combination of two sentences, or even implicit. We rely on a set of heuristics based on lexico-syntactic patterns and Wikipedia markup conventions in order to extract such sentences. A useful Wikipedia convention is the use of bold characters for visualizing the name of the referred entity in the page abstract: for example consider the Wikipedia page referring to "Vladimir Kramnik"[13] and the first paragraph of its abstract, depicted in Figure 2. Let us represent such paragraph as a sequence of n sentences $\{s_1, ..., s_n\}$. Typically, the bold words referring to the entity (*bold-name*) are included in a sentence $s_i, (i = 1, ..., n)$ that provides its definition according to a syntactic form of the type: "*bold–name <copula><predicative nominal||predicative adjective>*" (where *<copula>* is usually a form of the verb *to be*) e.g., "**Vladimir Borisovich Kramnik** is a Russian chess grandmaster". However, this is not always the case: sometimes, the sentence s_i containing the *bold-name* does not include any *<copula>*, while a *<copula>* can be found together with a co-reference to the entity, in one of the following sentences s_j. In such cases, we extract the entity definition by concatenating these two sentences i.e. $s_i + s_j$. If the abstract does not contain any *bold-name*, we inspect s_1: if it contains a *<copula>* we return s_1, otherwise we concatenate s_1 with the first of the next sentences e.g., s_i, containing a *<copula>* (i.e. $s_1 + s_i$). If none of the above is satisfied, we return s_1. We also apply additional heuristics for dealing with parentheses, and other punctuation. For the example in Figure 2 we return s_1: *Vladimir Borisovich Kramnik is a Russian chess grandmaster*, which contains the *bold-name* as well as a *<copula>*.

2. Natural language deep parsing of entity definitions (FRED). Once the entity definition has been extracted, it should be parsed and represented in a logical form that includes a set of types. In order to accomplish this task we use

[13] http://en.wikipedia.org/wiki/Vladimir_Kramnik

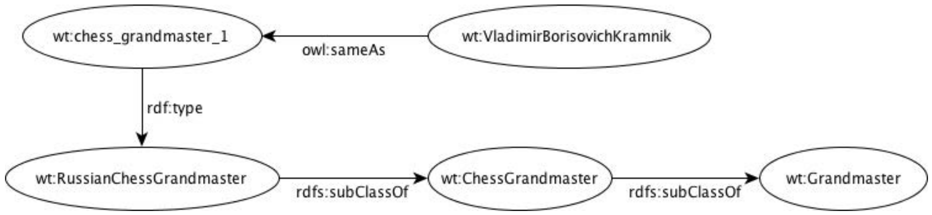


Fig. 3. FRED result for the definition "Vladimir Borisovich Kramnik is a Russian chess grandmaster"

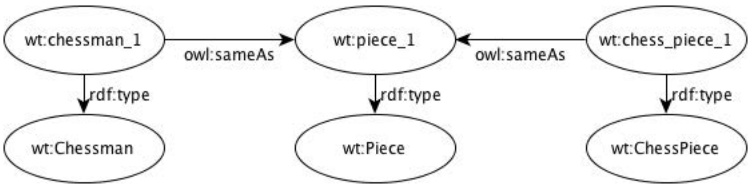


Fig. 4. FRED result for the definition "Chess pieces, or chessmen, are the pieces deployed on a chessboard to play the game of chess"

FRED[14], a tool that we have presented in [16]. It performs ontology learning by relying on Boxer [4], which implements *computational semantics*, a deep parsing method that produces a logical representation of NL sentences in DRT. FRED implements an alignment model and a set of heuristics for transforming DRT representations to RDF and OWL representations. In the context of this work, FRED is in charge of "reading" an entity NL definition, and producing its OWL representation, including a taxonomy of types. For example, given the above definition for **Vladimir Kramnik**, FRED returns the OWL graph depicted in Figure 3, containing the following taxonomy[15].

```
wt:RussianChessGrandmaster rdfs:subClassOf wt:ChessGrandmaster
                wt:ChessGrandmaster rdfs:subClassOf wt:Grandmaster
```

3. Selection of types and type-relations from the OWL graph (type selector). This step requires to identify, in FRED output graph, the paths providing typing information about the analyzed entity, and to discard the rest. Furthermore, we want to distinguish the case of an entity that is represented as an individual e.g. **Vladimir Kramnik**, from the case of an entity that is more appropriately represented as a class e.g., **Chess piece**. FRED output looks differently in these two situations as well as depending on the type of definition e.g., including a *copula* or parenthetic terms. For example, consider

[14] FRED is available online at http://wit.istc.cnr.it/stlab-tools/fred
[15] wt: http://www.ontologydesignpatterns.org/ont/wikipedia/type/

Table 1. Graph patterns and their associated type inferred triples for individual entities. Order reflects priority of detection. `[r]` $\in R =$ {wt:speciesOf, wt:nameOf, wt:kindOf, wt:varietyOf, w:typeOf, wt:qtyOf, wt:genreOf, wt:seriesOf}); [anyP] $\in \{*\} - R$.

ID	graph pattern (GP)	inferred axioms
gp_1	e `owl:sameAs` x && x `domain:aliasOf` y && y `owl:sameAs` z && z `rdf:type` C	e `rdf:type` C
gp_2	e `rdf:type` x && x `owl:sameAs` y && y `domain:aliasOf` z && w `owl:sameAs` z && w `rdf:type` C	e `rdf:type` C
gp_3	e `owl:sameAs` x && x `[r]` y && y `rdf:type` C	e `rdf:type` C
gp_4	e `owl:sameAs` x && x `rdf:type` C	e `rdf:type` C
gp_5	e `dul:associatedWith` x && x `rdf:type` C	e `rdf:type` C
gp_6	(e `owl:sameAs` x && x `anyP` y && y `rdf:type` C) || (e `anyP` x && x `rdf:type` C)	e `rdf:type` C

Table 2. Graph patterns and their associated type inferred triples for class entities. `[r]` $\in R =$ {wt:speciesOf, wt:nameOf, wt:kindOf, wt:varietyOf, w:typeOf, wt:qtyOf, wt:genreOf, wt:seriesOf}); [anyP] $\in \{*\} - R$.

ID	graph pattern (GP)	inferred axioms
gp_7	x `rdf:type` e && x `owl:sameAs` y && y `[r]` z && z `rdf:type` C	e `rdfs:subClassOf` C
gp_8	x `rdf:type` e && x `owl:sameAs` y && y `rdf:type` C	e `rdfs:subClassOf` C
gp_9	x `rdf:type` e && e `dul:associatedWith` y && y `rdf:type` C	e `rdfs:subClassOf` C
gp_{10}	(x `rdf:type` e && x `owl:sameAs` y && y `[anyP]` z && z `rdf:type` C) || (x `rdf:type` e && y `[anyP]` x && y `rdf:type` C)	e `rdfs:subClassOf` C

the entity **Chess piece**, which is a class entity, and is defined by *"Chess pieces, or chessmen, are the pieces deployed on a chessboard to play the game of chess."*. FRED output graph for such definition is depicted in Figure 4[16]. In this case, the graph paths encoding typing information comply with a different pattern from the one in Figure 3. The role of the type selector is to recognize a set of graph patterns that allow to distinguish between an entity being a class or an individual, and to select the concepts to include in its graph of types.

To implement the type selector, we have identified a set of graph patterns (GP), and defined their associated heuristics by following similar criteria as lexico-syntactic patterns [9], extended with the exploitation of RDF graph topology and OWL semantics. Currently, we use 10 GPs: 4 of them identifying class entities, and 6 for individual entities. Firstly, the type selector distinguishes if an entity is either an individual or a class entity: given an entity e, it is an individual if it participates in a graph pattern of type e `owl:sameAs` x, it is a class if it participates in a graph pattern of type x `rdf:type` e. As empirically observed, these two situations are mutually exclusive. After performing this distinction, the type selector follows a priority order for GP detection and executes the heuristics associated with the first matching GP. Tables 1 and 2 respectively report the GP sets and their associated heuristics by following the priority order used for detection, for individual entities and class entities.

[16] For space reasons, we include only the portion of the graph of interest in this context. Readers interested in visualizing the complete graph can submit the sentence to FRED online `http://wit.istc.cnr.it/stlab-tools/fred`.

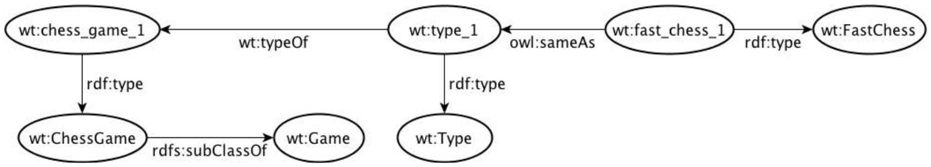


Fig. 5. FRED result for the definition "Fast chess is a type of chess game in which each side is given less time to make their moves than under the normal tournament time controls of 60 to 180 minutes per player"

Table 3. Normalized frequency of GPs on a sample set of ∼800 randomly selected Wikipedia entities

GP	frequency (%)	GP	frequency (%)	GP	frequency (%)	GP	frequency (%)	GP	frequency (%)
gp_1	0	gp_2	0.15	gp_3	3.98	gp_4	79.34	gp_5	0
gp_6	0.31	gp_7	1.11	gp_8	11.46	gp_9	0	gp_{10}	3.65

The rationale behind GP priority order resides in ontology design choices as well as in the way the current implementation of the type selector works. Sometimes, an entity definition from Wikipedia includes typing information from a "domain-level" as well as a "meta-level" perspective. For example, from the definition[17] *"Fast chess is a type of chess game in which each side is given less time to make their moves than under the normal tournament time controls of 60 to 180 minutes per player."* we can derive that "Fast chess" is a *type* (meta-level type) as well as a *chess game* (domain-level type). This situation makes FRED output include a GP detecting "type" as a type i.e., gp_8, as well as a GP detecting "chess game" as a type i.e., gp_7, as depicted in Figure 5. In this version of Tìpalo our goal is to type DBpedia entities only from a domain-level perspective. Furthermore, in its current implementation, the type selector executes only one heuristics: that associated with the first GP that matches in FRED output graph. Given the above rationale, gp_7 is inspected before gp_8. The same rationale applies to GP for individual entities, illustrated in Table 1.

For the `dbp:Fast_chess`[18] example, the type selector detects that the entity is a class and the first GP detected is gp_7, hence it produces the additional triples:

```
dbp:Fast_chess rdfs:subClassOf wt:ChessGame
      wt:ChessGame rdfs:subClassOf wt:Game
```

The execution of Tìpalo pipeline on a sample set of randomly selected ∼800 Wikipedia entities[19] has shown that the most frequent GPs are gp_4 and gp_8, which is not surprising, since they correspond to the most common linguistic patterns for definitions. Table 3 reports the frequency of each GP on the sample set.

[17] `http://en.wikipedia.org/wiki/Fast_chess`
[18] dbp: `http://dbpedia.org/resource/`
[19] Details about the definition of the sample set are given in Section 5.

The *type selector* implements an additional heuristics: it detects if any of the terms referring to a type in the graph can be referenceable as a DBpedia entity. For example, the term "chess" in the definition of "Fast chess" is resolvable to `dbp:Chess`. In such case, the type selector produces the following triple:

```
dbp:Fast_chess rdfs:subClassOf dbp:Chess
```

This additional heuristics improves the internal linking within DBpedia, resulting in higher cohesion of the resource graph.

By following the defined heuristics, we are able to select the terms that refer to the types of an entity e, and to create a namespace of Wikipedia types that captures the variety of terms used in Wikipedia definitions[20].

4. Word sense disambiguation engine (UKB). After having identified the concepts expressing the types of an entity and their taxonomical relations, we have to gather their correct sense: we need a WSD tool. A possible way to achieve this goal is to identify alignments between the type terms and WordNet terms. We have approached this issue by applying two alternative solutions on a sample set of 100 entity definitions. The first approach involves UKB [1], a tool which returns the WordNet synset for a term, looking for the one that fits best the context given by the entity definition. UKB provids good results in terms of precision and recall although its speed performance needs improvement in order to apply it on a large dataset such as Wikipedia. We have plans for dealing with this issue in the next future (cf. Secion 5.1). The second solution is to select the most frequent WordNet sense for a given term, which is very efficient in terms of speed, but shows lower precision and recall[21]. This step allows us to assign a WordNet type (corresponding to the identified synset) to an entity. Referring to the above example (i.e., definition of fast chess), we produce the following additional triples[22]:

```
wt:ChessGame owl:equivalentTo wn30syn:synset-chess-noun-2
    wt:Game owl:equivalentTo wn30syn:synset-game-noun-1
```

5. Identifying other Semantic Web types (type matcher). So far the typing process produces a set of newly defined concepts, and disambiguates them to a WordNet sense. The final step consists in linking such concepts to other Semantic Web ontologies, in order to support shared interpretation and linked data enrichment. In order to exemplify this task with respect to the goal of gathering top-level types[23], we have produced and published two RDF

[20] Wikipedia class taxonomy, wt: = `http://www.ontologydesignpatterns.org/ont/wikipedia/type/`

[21] The current implementation of Tìpalo relies on UKB for word sense disambiguation.

[22] wn30syn: = `http://purl.org/vocabularies/princeton/wn30/instances/`

[23] Any other aligned ontology, or ontology matching component can be used in order to perform arbitrary type matching.

datasets (see Section 3) containing alignments between WordNet synsets and Super Senses (broad lexico-semantic categories), and between WordNet synsets and some foundational ontology classes. The *type matcher* exploits these alignments in order to produce additional `rdf:type` triples. For example, for the entity `dbp:Fast_chess`, the type matcher produces e.g. the following triples:

```
        wt:ChessGame rdfs:subClassOf d0:Activity
wt:Game rdfs:subClassOf wn30:supersense-noun_act
```

meaning that the term "chess game" associated with the WordNet sense `wn30syn:synset-chess-noun-2` (as provided by the WSD component) is aligned to the class `Activity` of Dolce Zero[24] ontology. Analogously, the term "game" with its sense `wn30syn:synset-game-noun-1` is aligned to the WordNet super sense "act".

The described five steps compose the Tìpalo automatic typing procedure, whose output feeds incrementally a Wikipedia ontology based on the entity definitions provided by the crowds, hence able to reflect the richness of natural language and with a potentially complete domain coverage. The Wikipedia ontology is encoded in two semantic resources i.e., the Wikipedia entity types dataset and the Wikipedia class taxonomy dataset, which are described in Section 5.

5 Results and Evaluation

In this section we report the results of our work, and evaluate them in terms of precision, recall and time of computation. Our final goal is to incrementally build a *Wikipedia ontology* based on how users describe things, hence able to capture the richness of NL definitions. To this aim, we have developed a web service, called *Tìpalo*, which, given a description of an entity, produces a RDF named graph describing its typing information by means of DBpedia entities, WordNet synsets and supersenses, and foundational ontology classes. Besides the RDF resources described in Section 3, and the methods in Section 4, we have developed a novel resource and two evaluation tools.

Wikipedia ontology. We have produced a demonstrating Wikipedia ontology[25] by analyzing a randomly selected sample of 800 Wikipedia pages. The resulting Wikipedia ontology consists of two RDF datasets, one containing `rdf:type` triples defining DBpedia entity types, and another containing ontology classes related by means of `rdfs:subClassOf` and `owl:equivalentTo` axioms. The two datasets can be queried through a SPARQL endpoint (or downloaded as dump files) either as a whole graph or as separated named graphs, each associated with a single entity. Each named graph has an ID starting with `dbpedia_` followed by a DBpedia entity local name e.g., `dbpedia_Vladimir_Kramnik`.

[24] Dolce Zero, `http://www.ontologydesignpatterns.org/ont/d0.owl`

[25] Link available at the Wikipedia ontology page: `http://stlab.istc.cnr.it/stlab/WikipediaOntology/`

Reusable evaluation tools. Additionally, we have developed two tools for evaluating our method, one for collaboratively building a golden standard, and the other for evaluating the Wikipedia ontology (both tools are described in Section 5.1).

5.1 Evaluation

We evaluate our work considering the accuracy of types assigned to the sample set of Wikipedia entities, and the soundness of the induced taxonomy of types for each DBpedia entity. The accuracy of types has been measured in two ways: (i) in terms of precision and recall against a golden standard of 100 entities, and (ii) by performing a user study. The soundness of the induced taxonomies has been assessed in a user study.

Building a sample set of Wikipedia pages. We have performed our experiments on a sample set of ∼800 randomly selected Wikipedia pages. From the 800 set, we have removed all pages without an abstract text, e.g. *redirect* pages, categories, and images. The resulting sample includes 627 pages with the following characteristics: (i) each page has a corresponding DBpedia entity, (ii) each DBpedia entity has a DBpedia type, a YAGO type, or no type, (iii) 67.62% of the corresponding DBpedia entities have a YAGO type, 15.47% have a DBPO type, and 30% of them have no type.

Building a golden standard. We have built a manually annotated golden standard of Wikipedia entity types based on the sample set used for our experiments. To support this process we have developed a web-based tool named *WikipediaGold*[26] that manages argumentation among users in order to support them in discussing and reaching agreement on decisions (agreement was considered reached with at least 70% users giving the same answer). Ten users with expertise in ontology design (four senior researchers and six PhD students in the area of knowledge engineering) have participated in this task, and have reached agreement on 100 entities. We have used such 100 entities as a golden standard for evaluating and tuning our method. The golden standard can be retrieved from the cited Wikipedia Ontology page, and it can be useful for future development and for comparing our work with possible other approaches to this same task.

WikipediaGold is based on a simple user task, repeated iteratively: given an entity e e.g., `dbp:Vladimir_Ramnik`, WikipediaGold visualizes its definition e.g., *"Vladimir Borisovich Kramnik is a Russian chess grandmaster."* and asks users to:

- indicate if e refers to a concept/type or to a specific instance. Users can select either "is a" or "is a type of" as possible answers. This value allows us to evaluate if our process is able to distinguish entities which are typical individuals, from those that are typical classes;

[26] Available online at `http://wit.istc.cnr.it/WikipediaGold`, demonstrating video at `http://wit.istc.cnr.it/stlab-tools/video/`

Table 4. Performance evaluation of the individual pipeline components

Component	precision	recall	F-measure (F1)
Type selector	.93	.90	.92
WSD (UKB)	.86	.82	.84
WSD (most frequent sense)	.77	.73	.75
Type matcher (Supersense)	.73	.73	.73
Type matcher (DUL+/D0)	.80	.80	.80

- copy and paste the terms in the definition that identifies the types of e, or indicate a custom one, if the definition does not contain any. In our example, a user could copy the term *"Russian chess grandmaster"*. This information is meant to allow us evaluating the performances of the *type selector*;
- select the most appropriate concepts for classifying e from two list of concepts. The first list includes 21 WordNet supersenses, and the second list includes 23 classes from DULplus and Dolce Zero. Each concept is accompanied by a describing gloss and some examples to inform the user about its intended meaning. In the example, users can select the type "Person" available in both lists. The two lists of concepts are available online at the Wikipedia ontology page.

For each answer, users can optionally include a comment motivating their choice. When there is disagreement among users about an entity, WikipediaGold submits it again to users who have already analyzed it. In these cases a user can see other users' choices and comments, and decide if either to keep her decision, or to change it. In both cases, a comment motivating own decision must be entered.

Evaluation against the golden standard. Our evaluation is based on measuring precision and recall of the output of the three main steps of the process, against the golden standard: (i) type selection (step 3), (ii) word sense disambiguation (WSD) (step 4), and (iii) type matching (step 5). We also measure precision and recall of the overall process output.

Table 5. Performance evaluation of the overall process

Typing process	precision	recall	F-measure (F1)
WordNet types	.76	.74	.75
Supersenses	.62	.60	.61
Dul+/D0	.68	.66	.67

The results shown in Table 4 indicate the performances of the individual components. The type selector stands out as the most reliable component ($F1 = .92$), which confirms our hypothesis that a rich formalization of definitions and a good design of graph patterns are a healthy approach to entity typing. The WSD task has been performed with two approaches: we analyze its performance by executing UKB as well as a most-frequent-sense-based (MFS) approach. UKB shows to perform better ($F1 = .84$) than MFS ($F1 = .75$), suggesting that Wikipedia definitions often include polysemous senses, and that the used language tends to be specialized i.e., polysemous terms are used with different senses. The type matcher performs better with DULplus/Dolce Zero types than with WordNet supersenses, which shows an improvement with respect to the state of the art considering that WordNet super senses are considered an established and reliable semantic resource when used as a top-level ontology for WordNet.

Table 5 illustrates the performance of the overall automatic typing process. As expected, the steps that map the extracted types to WordNet types, super senses, and top-level ontologies tend to decrease the initial high precision and recall of the type selector. In fact, when put into a pipeline, errors typically reinforce previous ones, producing in this case an overall decrease of $F1$ from .92 of the type selection step to .75 of the combined type selection and WSD, to .67 with the addition of DULplus/Dolce Zero alignment (type matcher). However, the modularity of our process enables the reuse of results that are actually useful to a certain project, e.g. discarding a step that performs worse.

The good performances observed in our evaluation experiments make us claim that using Tìpalo brings advantages when compared to the most prominent existing approaches i.e., DBpedia project [11] and YAGO [17] to DBpedia entity typing, for the following reasons: (i) Tìpalo potentially ensures complete coverage of Wikipedia domains (intensional coverage) as it is able to capture the reachness of terminology in NL definitions and to reflect it in the resulting ontology, while DBpedia and YAGO depend both on the limited intensional completeness of infobox templates and Wikipedia categories, (ii) Tìpalo is independent from the availability of structured information such as infobox templates and Wikipedia categories, hence ensuring higher extensional completeness as most Wikipedia entities have a definition while many of them lack infoboxes.

A direct comparison of our results with DBpedia and YAGO approaches occurred to be unfeasible in the scope of this paper because the two approaches differ from ours on important aspects: they use different reference type systems; they rely on Wikipedia categories or infobox templates while we rely on the NL descriptions used for defining Wikipedia entities by the crowds, hence it is difficult (if not impossible) to compare the derived vocabularies; finally, the granularity of their type assignments is heterogeneous. These cases make it hard to define criteria for performing a comparison between the accuracy of the automatically assigned types. Hence, we could not consider either DBpedia or YAGO suitable golden standards for this specific task, which motivates the construction of a specific golden standard.

Evaluation by user study. In order to further verify our results, we have conducted a user study. We have implemented a second Web-based tool, named *WikipediaTypeChecker*[27], for supporting users in expressing their judgement on

[27] Available online at `http://wit.istc.cnr.it/WikipediaTypeChecker`, demonstrating video at `http://wit.istc.cnr.it/stlab-tools/video`

the accuracy of Tìpalo types assigned to the sample set of Wikipedia entities. WikipediaTypeChecker is available online.

WikipediaTypeChecker asks users to evaluate the accuracy of Tìpalo types, the soundness of the induced taxonomies, and the correctness of the selected meaning of types, by expressing a judgement on a three-valued scale: *yes, maybe, no.* Users' task, given an entity with its definition, consists of three evaluation steps. Consider for example the entity `dbp:Fast_chess`: in the first step, users evaluate the accuracy of the assigned types by indicating the level of correctness of proposed types. In this example, for the entity "Fast chess" three types are proposed: "Chess game", "Game", and "Activity"; in the second step users validate the soundness of the induced taxonomy of types for an entity. In this example, the proposed taxonomy is `wt:ChessGame rdfs:subClassOf wt:Game`; in the third step users evaluate the correctness of the meaning of individual types (i.e. WSD). For example, the proposed meaning for "Chess game" is "a board game for two players who move their 16 pieces according to specific rules; the object is to checkmate the opponent's king". Five users with expertise in knowledge engineering have participated in the user study (three PhD students and two senior researchers). For each entity and for each evaluation step, we have computed the average value of judgements normalized to an interval [0,1], which gives us a value for the precision of results. The results are shown in Table 6, with a (high) inter-rater agreement (Kendall's W) of .79[28].

Table 6. Results of the user-based evaluation, values indicate precision of results. Inter-rater agreement (Kendall's W) is .79, Kendall's W ranges from 0 (no agreement) to 1 (complete agreement).

Task	Type extraction	Taxonomy induction	WSD
Correctness	.84	.96	.81

These results confirm those observed in the evaluation against a golden standard (cf. Tables 4 and 5). In this case, we have split the evaluation of the correctness of extracted types between *assigned types* (.84), and *induced taxonomy* (.96): their combination is comparable to the precision value observed for the type selector against the golden standard (.93). The performance of the WSD task is a bit lower (.81 against .86 precision), which suggests the need for additional evaluation of WSD performance, and exploration of possible alternative solutions.

Estimating time performance. In the scope of this work, we could only perform a preliminary estimation of time performances, since we have run the process on simple desktop machines. The workflow process and storage data

[28] Kendall's W is a coefficient of concordance used for assessing agreement among raters. It ranges from 0 (no agreement) to 1 (complete agreement), and is particularly suited in this case as it makes no assumptions regarding the nature of the probability distribution and handles any number of distinct outcomes.

have been executed on an Intel Pentium DualCore 2.80GHz with 1GB RAM, while UKB and FRED run on a Quad-Core Intel Xeon 2,26 GHz RAM 32 GB Processor Interconnect Speed: 5.86 GT/s. With this setting, the whole process takes maximum ∼11 seconds per definition (depending on its complexity). Type selection and top-level matching are instantaneous, while there is a bottleneck due to UKB performance: our process become extremely fast (maximum ∼2 seconds per definition) if we remove UKB and disambiguate terms by selecting synsets with the most-frequent-sense-based method (with some degradation in precision and recall). We remark that in this implementation UKB and FRED are used as web services and remotely invoked (one machine is in Rome and another is in Bologna), hence suffering from delay due to network latency. We are confident that by parallelizing the whole process on a more robust cluster, and deploying all components and data sources locally, the speed will significantly increase[29], which reasonably suggest the applicability of the process on a large-scale dataset such as Wikipedia.

6 Conclusions and Future Work

We have presented Tìpalo, an implemented method that formalizes entity definitions extracted from Wikipedia for automatically typing DBpedia entities and linking them to other DBpedia resources, WordNet, and foundational ontologies. We have experimented Tìpalo on a sample set of ∼800 Wikipedia entities. Results have been evaluated against a golden standard and by a user study, and are up to the task, specially for the pure type selection task. In ongoing work, we are deploying the tool on a more robust cluster for improving time performances and experimenting on a large-scale resource such as the whole Wikipedia. The medium-term goal is to incrementally build a Wikipedia ontology that reflects the richness of terminology expressed by natural language, crowd sourced definitions of entities.

References

1. Agirre, E., Soroa, A.: Personalizing pagerank for word sense disambiguation. In: Proceedings of the 12th Conference of the European chapter of the Association for Computational Linguistics (EACL 2009), Athens, Greece. The Association for Computer Linguistics (2009)
2. Cimiano, P.: Ontology Learning and Population from Text: Algorithms, Evaluation and Applications. Springer (2006)
3. Cimiano, P., Völker, J.: Text2onto - a framework for ontology learning and data-driven change discovery (2005)
4. Curran, J.R., Clark, S., Bos, J.: Linguistically motivated large-scale nlp with c&c and boxer. In: Proceedings of the ACL 2007 Demo and Poster Sessions, Prague, Czech Republic, pp. 33–36 (2007)

[29] We have plans to perform additional tests in such an environment in the immediate future.

5. Etzioni, O., Fader, A., Christensen, J., Soderland, S., Mausam: Open information extraction: The second generation. In: IJCAI, pp. 3–10. IJCAI/AAAI (2011)
6. Gangemi, A.: Norms and plans as unification criteria for social collectives. Autonomous Agents and Multi-Agent Systems 17(1), 70–112 (2008)
7. Gangemi, A., Navigli, R., Velardi, P.: The OntoWordNet Project: Extension and Axiomatization of Conceptual Relations in WordNet. In: Meersman, R., Schmidt, D.C. (eds.) CoopIS/DOA/ODBASE 2003. LNCS, vol. 2888, pp. 820–838. Springer, Heidelberg (2003)
8. Hartmann, S., Szarvas, G., Gurevych, I.: Mining multiword terms from wikipedia. In: Pazienza, M.T., Stellato, A. (eds.) Semi-Automatic Ontology Development: Processes and Resources, pp. 226–258. IGI Global, Hershey (2012)
9. Hearst, M.A.: Automatic acquisition of hyponyms from large text corpora. In: COLING, pp. 539–545 (1992)
10. 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)
11. Lehmann, J., Bizer, C., 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)
12. Navigli, R.: Word sense disambiguation: A survey. ACM Comput. Surv. 41(2) (2009)
13. Navigli, R., Ponzetto, S.P.: BabelNet: Building a very large multilingual semantic network. In: Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, Uppsala, Sweden, July 11-16, pp. 216–225 (2010)
14. 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)
15. Nuzzolese, A.G., Gangemi, A., Presutti, V., Ciancarini, P.: Type inference through the analysis of wikipedia links. In: WWW 2012 Workshop on Linked Data on the Web (LDOW 2012). CEUR (2012)
16. Presutti, V., Draicchio, F., Gangemi, A.: Knowledge Extraction Based on Discourse Representation Theory and Linguistic Frames. 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. 114–129. Springer, Heidelberg (2012)
17. Suchanek, F.M., Kasneci, G., Weikum, G.: Yago: A Core of Semantic Knowledge. In: 16th International World Wide Web Conference (WWW 2007). ACM Press, New York (2007)
18. Tanev, H., Magnini, B.: Weakly supervised approaches for ontology population. In: Proceedings of the 2008 Conference on Ontology Learning and Population: Bridging the Gap between Text and Knowledge, pp. 129–143. IOS Press, Amsterdam (2008)
19. Völker, J., Rudolph, S.: Lexico-logical acquisition of owl dl axioms – an integrated approach to ontology refinement (2008)
20. Witte, R., Khamis, N., Rilling, J.: Flexible ontology population from text: The owlexporter. In: Calzolari, N., Choukri, K., Maegaard, B., Mariani, J., Odijk, J., Piperidis, S., Rosner, M., Tapias, D. (eds.) LREC. European Language Resources Association (2010)

Performance Heterogeneity and Approximate Reasoning in Description Logic Ontologies

Rafael S. Gonçalves, Bijan Parsia, and Ulrike Sattler

School of Computer Science
University of Manchester
Manchester, United Kingdom

Abstract. Due to the high worst case complexity of the core reasoning problem for the expressive profiles of OWL 2, ontology engineers are often surprised and confused by the performance behaviour of reasoners on their ontologies. Even very experienced modellers with a sophisticated grasp of reasoning algorithms do not have a good mental model of reasoner performance behaviour. Seemingly innocuous changes to an OWL ontology can degrade classification time from instantaneous to too long to wait for. Similarly, switching reasoners (e.g., to take advantage of specific features) can result in wildly different classification times. In this paper we investigate performance variability phenomena in OWL ontologies, and present methods to identify subsets of an ontology which are performance-degrading for a given reasoner. When such (ideally small) subsets are removed from an ontology, and the remainder is much easier for the given reasoner to reason over, we designate them "hot spots". The identification of these hot spots allows users to isolate difficult portions of the ontology in a principled and systematic way. Moreover, we devise and compare various methods for approximate reasoning and knowledge compilation based on hot spots. We verify our techniques with a select set of varyingly difficult ontologies from the NCBO BioPortal, and were able to, firstly, successfully identify performance hot spots against the major freely available DL reasoners, and, secondly, significantly improve classification time using approximate reasoning based on hot spots.

1 Introduction

Reasoning tasks on ontologies expressed in a rich description logic such as that underlying OWL 2 have a high worst case complexity. As a consequence, reasoning time can be highly unpredictable: seemingly innocuous changes to an ontology might shift reasoning time from seconds to days; different reasoners might have wildly different behaviour on the same input. Even seasoned reasoner developers do not have a mental performance model sufficient to deal with many, particularly novel, cases (indeed, this fact keeps reasoner optimisation research a lively area).

Mere high worst case complexity, of course, does not entail unpredictability. The difficulty of determining the satisfiability of propositional k-CNF formulae (the k-SAT problem), for example, is highly predictable by attending to the "density" (i.e., the ratio of number of clauses to number of distinct variables) of a formula. Not only is it predictable, but there is an increasingly sophisticated theoretical understanding of this behaviour. This predictability has been observed in various modal logics which correspond to the description logics commonly used as ontology languages [14,10]. However, several observations belie the utility of these results: 1) Even for comparatively simple logics such as $\mathcal{ALC}$ the number of parameters becomes unwieldy: while propositional logic has two main parameters (for a given size, k!) — number of clauses (L) and number of variables (N) — $\mathcal{ALC}$ adds (at least) modal depth (d), the number of roles (i.e., modalities, m), and the proportion of modal to propositional atoms [10]. 2) The inputs are highly regimented and bear little relationship to the sorts of formulae found in practice, especially in manually crafted artifacts such as ontologies. For example, all ontologies have axioms, not just concept expressions, these axioms often "break up" complex concepts, and reasoners exploit this fact.[1] Thus, to predict behaviour of realistic or naturally occurring ontologies, we need to understand even more parameters (perhaps dozens), and normalizing away that complexity is unlikely to be helpful. 3) Reasoners have different suites of optimizations and even underlying calculi, thus respond differently to these inputs.

Together, these observations suggest that users crafting ontologies are likely to be surprised[2] by the enormous variation in performance behaviour which does not relate intuitively to the changes they make (either in the ontology or in the reasoner used). Three basic phenomena startle users: 1) An ontology which takes seconds to classify[3] in one reasoner, effectively fails to terminate with another. 2) Ontologies of similar size and apparent complexity take wildly different times to classify on the same reasoner. 3) Apparently innocuous changes to a single ontology result in large increases (or decreases) in classification time.[4] Of course, the primary negative phenomenon is excessive reasoning time.

The most prominent, principled way to cope with this problem is to shift to a less expressive logic, such as OWL $\mathcal{EL}$, for which classification is decidable in polynomial time. Reasoning in $\mathcal{EL}$ (and similar logics) is not only polynomial (in general) but has proven to be rather robust to novel input [1,2]. This move is not always possible, as it involves severe limitations on what can be expressed. Similarly, approximate reasoning (i.e., giving up on soundness or completeness) can make reasoning performance significantly better and more predictable, but at the cost of increased uncertainty about the results [18,16,15]. In practice, users often modify their ontologies based on folk wisdom ("negation is hard", "inverses are hard"), on bespoke advice from reasoner developers, or randomly.

[1] "In realistic KBs, at least those manually constructed, large and complex concepts are seldom described monolithically, but are built up from a hierarchy of named concepts whose descriptions are less complex." [9]

[2] "[Reasoner] performance can be scary, so much so, that we cannot deploy the technology in our products." — Michael Shepard
`http://lists.w3.org/Archives/Public/public-owl-dev/2007JanMar/0047.html`

[3] Throughout, we focus on *classification* as the key reasoning task, as it is the most prevalent service invoked by ontology developers.

[4] Esp. distressing are removals that increase time, and additions which decrease it dramatically.

We need a better understanding of reasoning performance variability, or at least methodologies for analyzing it in particular cases. The contributions of this paper are as follows: for an ontology $\mathcal{O}$ that a reasoner R takes 'too long' to classify, we have *designed and thoroughly evaluated* (1) *a technique for analyzing the performance variability of R on $\mathcal{O}$*, (2) *a technique to isolate subsets of $\mathcal{O}$ that contribute negatively to R's high classification time, so called* hot spots, and (3) *a series of techniques to approximate the hot spot in $\mathcal{O}$.*

Firstly, we have verified, via technique (1), that there exist two kinds of performance profiles; an ontology-reasoner pair can be performance "heterogeneous" or performance "homogeneous", depending on whether there are certain kinds of performance variability between subsets of the ontology. Secondly, we identified very small subsets of an ontology whose removal causes a significant decrease in classification time, i.e., hot spots, using technique (2). Indeed we show that performance heterogeneous ontology-reasoner pairs are highly likely to have such subsets which are detectable by our methods. Thirdly, and finally, we show that if there is a hot spot for an ontology-reasoner pair, then we can approximate it in such a way that our criteria for a hot spot (i.e., classification time boost and size) are maintained.

2 Preliminaries

We assume the reader to be reasonably familiar with ontologies and OWL [22], as well as the underlying description logics (DLs) [8]. An ontology $\mathcal{O}$ is a set of axioms, and its *signature* (the set of individuals, concept and role names used) is denoted $\widetilde{\mathcal{O}}$. We use the notion of a *locality-based module* [5], which is a subset of an ontology $\mathcal{O}$ that preserves all consequences of $\mathcal{O}$ w.r.t. to a signature Σ. An x-module $\mathcal{M}$ extracted from an ontology $\mathcal{O}$ for a signature Σ is denoted x-$mod(\Sigma, \mathcal{O})$, for x one of $\top\bot^*$, $\top$ or $\bot$. A justification $\mathcal{J}$ of a consequence α is a $\subseteq$-minimal subset of an ontology $\mathcal{O}$ that is sufficient for α to hold [11]. The reasoning time of an ontology $\mathcal{O}$ using reasoner R, denoted RT$(\mathcal{O}, R)$,[5] comprises the time for consistency checking, classification (computing atomic subsumptions) and coherence (concept satisfiability). The set of atomic subsumptions resulting from the classification of an ontology $\mathcal{O}$ is denoted $Cl(\mathcal{O})$.

3 Materials

In order to test our methods we need a reasonable corpus of "problem" ontologies. We derived one from the NCBO BioPortal, a large collection of user contributed, "working" ontologies covering a wide range of biomedical domains [13]. We gathered all ontologies from the BioPortal, and performed a reasoner performance test across this corpus. Four major, freely available DL reasoners were used: Pellet (v2.2.2) [20], HermiT (v1.3.6) [19], FaCT++ (v1.5.3) [21], and

[5] When R is clear from the context, we also use RT$(\mathcal{O})$.

JFact (v0.9).[6] The experiment machine is an Intel Xeon Quad-Core 3.20GHz, with 32GB DDR3 RAM dedicated to the Java Virtual Machine (JVM v1.5). The system runs Mac OS X 10.6.8, all tests were run using the OWL API v3.3 [7].

The entire BioPortal corpus contains 216 ontologies. We discarded all ontologies with reasoning times, for all reasoners, below 60 seconds (i.e., the "easy" ontologies). This leaves 13 ontologies, 3 of which did not classify within 10 hours: the IMGT[7] ontology with Pellet, GALEN[8] with all reasoners, and GO-Ext. (Gene Ontology Extension),[9] with FaCT++ and JFact.

The naive approach to determining heterogeneity is to enumerate the "acceptably" small subsets of the ontology and measure the classification time for each. Given that our ontologies range from 100s to over 100,000 axioms, this is obviously infeasible. Random testing of acceptably small subsets might be effective assuming that a sufficiently large proportion of those subsets were, in fact, hot spots, though our preliminary experiments in this direction were unpromising. Instead, we performed two sorts of heterogeneity detection. In the first, "coarse grained" method, we classify ontology-reasoner pairs as performance heterogeneous or homogenous by attending to performance fluctuations (or lack thereof) over relatively large, evenly increasing subsets of the ontology. In the second, we apply two simple heuristics for selecting candidate subsets, and then verify whether they conform to our hot spot criteria. The second method directly verifies our heterogeneity condition.

4 Coarse Grained Prediction

For current purposes, we focus on performance variability of a single reasoner for a given ontology. In particular, we are always examining the difference in reasoning time of select subsets of a given, hard-for-a-specific-reasoner ontology. We do this for several reasons: 1) it simulates a common user scenario (e.g., editing or trying to "optimize" an ontology) and 2) we are investigating the background assumption that ontologies which are difficult (e.g., approach the worst case) are often so in a "fragile" way, i.e., their performance is sensitive to small changes.

We say that an ontology is *performance homogenous* for a reasoner if there is a linear factor L and variable k such that for all $\mathcal{M} \subseteq \mathcal{O}$ and for $k \cdot |M| = |\mathcal{O}|$, we have that $L \cdot k \cdot \mathrm{RT}(\mathcal{M}) \approx \mathrm{RT}(\mathcal{O})$. An ontology which is not *performance homogeneous* we call *performance heterogeneous*. It is important to note that, in both cases, the performance profile of the ontology and its subsets may be *predictable* (even if we currently do not know how to predict it).

In this experiment, each ontology is divided into 4 and 8 random subsets of equal size, and the classification times of these subsets as increments are measured (i.e., we measure, for the 4-part division, $\mathrm{RT}(\mathcal{O}_1)$, $\mathrm{RT}(\mathcal{O}_1 \cup \mathcal{O}_2)$, $\mathrm{RT}(\mathcal{O}_1 \cup \mathcal{O}_2 \cup \mathcal{O}_3)$, $\mathrm{RT}(\mathcal{O})$, where $\mathcal{O}_1, \mathcal{O}_2, \mathcal{O}_3$ are subsets of $\mathcal{O}$). Both

6 http://jfact.sourceforge.net/

7 http://www.imgt.org/IMGTindex/ontology.html

8 http://www.co-ode.org/galen/

9 http://www.geneontology.org/

measurements are carried out several times per ontology (at least 10, though often more), where each time the list of axioms in $\mathcal{O}$ is shuffled.

Note that we are testing a very small number of subsets of each ontology, so, in principle, that we see "smooth" behaviour could be due to insufficient sampling. However, because each increment is rather large, we hope that it will contain (in a behaviour exhibiting way) any hot spots.

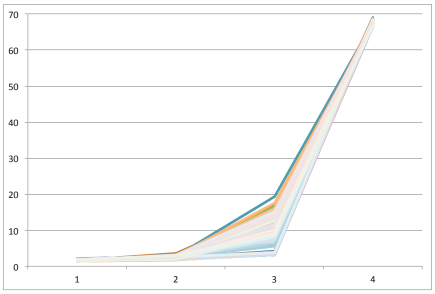
Overall 4 out of 13 ontology/reasoner pairs exhibit roughly linear performance growth in our tests (see Figures 1c and 1d for characteristic graphs). GALEN proved infeasible to work with (even only half the ontology gave reasoning times of over 10 hours), and was discarded. The remainder exhibited non-linear and sometimes highly variable performance behaviour. For example, Figure 1e shows that even the very coarse, 4-part division method can detect strange performance patterns, although the more fine grained, predictably, is more detailed (Figure 1f). Contrariwise, Figure 1a shows a rather smooth, if non-linear, curve. It is tempting to think that that smoothness indicates a relatively predictable performance profile, but as we see in the more fine grained view (Figure 1b) this is not true. However, this supports (though, obviously, does not confirm) our hypothesis that ontologies with non-linear growth, whether smooth or jaggy, are performance heterogeneous. In our corpus, they certainly exhibit surprising variability.

While we were unable to run this test sufficient enough times to attain statistical significance for all ontologies, the data gathered is already highly suggestive of reasoner performance behaviour on our test corpus. During the execution of this experiment we noted a curious phenomenon: While in most ontologies we managed to achieve convergence on the overall classification time on each run, in the GO-Ext ontology this did not happen. Surprisingly, the classification time of GO-Ext with Pellet, under exactly the same experimental conditions, varies from seconds to hours; more specifically, the range is from 27 seconds to 1 hour and 14 minutes (Figure 2). A unique case as it may be (in our corpus), it suffices to illustrate not only the need for performance analysis solutions, but also the difficulty of the problem in cases such as this one.

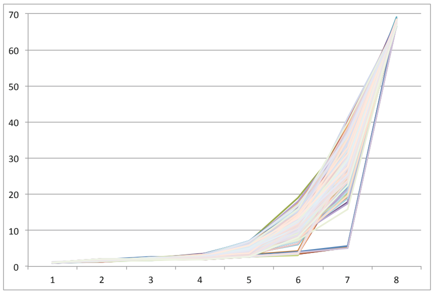
5 Performance Hot Spots

We hypothesise that if an ontology is *performance heterogeneous* for a reasoner, then there exists at least one "small" subset of that ontology whose removal results in a "significant" change in the classification time (positive or negative). That is, when there exists a subset $\mathcal{M}$ of a given ontology $\mathcal{O}$ such that (1) $\mathcal{M}$ is "acceptably small" (typically, $|\mathcal{M}| \ll |\mathcal{O}|$), and (2) $\mathrm{RT}(\mathcal{O} \setminus \mathcal{M}) \ll (\mathit{or} \gg) \mathrm{RT}(\mathcal{O})$. We call such a subset $\mathcal{M}$, which witnesses the performance heterogeneity of $\mathcal{O}$, a "hot spot", by way of analogy with program profilers. The analogy is imperfect as we cannot say whether such bits themselves consume an inordinate amount of time, or whether they have some more diffuse triggering effect.

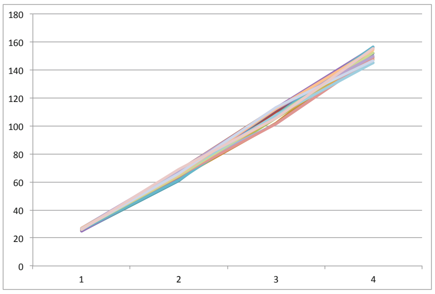
Obviously, the exact nature of the smallness of $\mathcal{M}$ relative to $\mathcal{O}$ and the respective classification times depend on non-intrinsic considerations. In general, we consider subsets below 20% of the ontology and speed-ups of at least an order of magnitude, and preferably more.



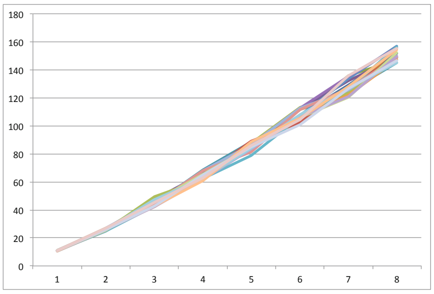
(a) ChEBI 4-part division (Pellet)



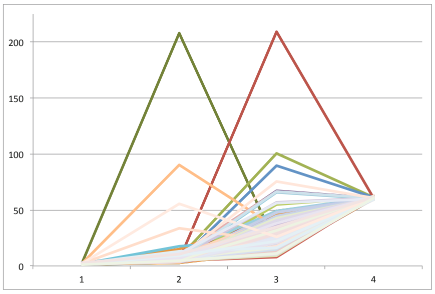
(b) ChEBI 8-part division (Pellet)



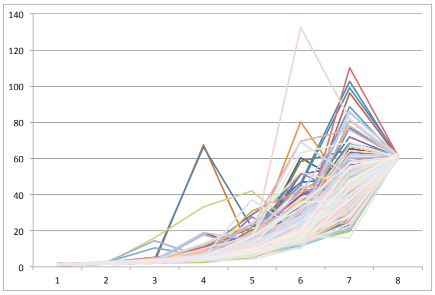
(c) Gazetteer 4-part division (HermiT)



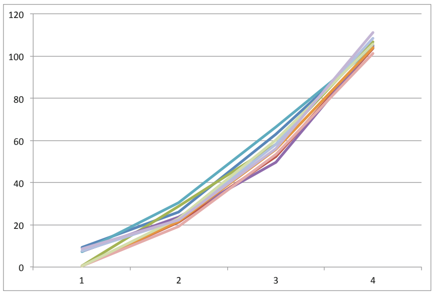
(d) Gazetteer 8-part division (HermiT)



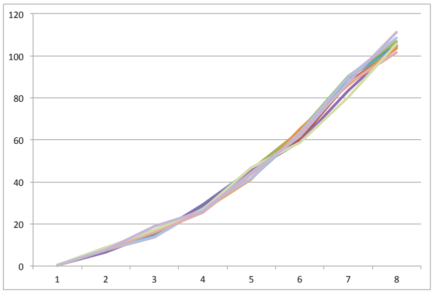
(e) EFO 4-part division (Pellet)



(f) EFO 8-part division (Pellet)

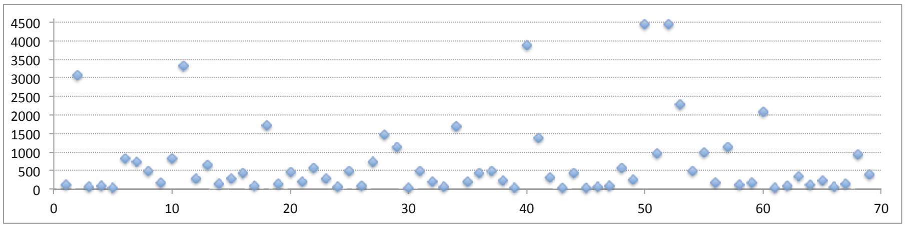


(g) ICF 4-part division (HermiT)

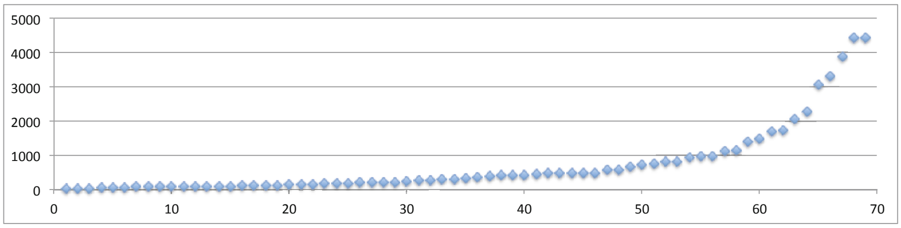


(h) ICF 8-part division (HermiT)

Fig. 1. Performance heterogeneity tests of select ontologies. All times in seconds.



(a) Times in chronological order.



(b) Times in ascending order.

Fig. 2. Classification times (in seconds) of the GO-Ext ontology with Pellet

Given that exhaustive search is unpromising, indeed the search space is unmanageable; for a number of axioms n, variable k, and considering only subsets of size below 20% of n, the possible subsets are all unique combinations of n of size k, for $1 \leqslant k \leqslant 0.2n$, we need some other method for producing good "candidate hot spots", i.e., subsets that are likely to be hot spots. In [23], the authors suggest that the satisfiability checking (SAT) time of an atomic concept is an indicator of the total time the reasoner spends on or "around" those atomic concepts during classification. In particular, they observe that in their examined ontologies, relatively few concepts (2-10 out of 1000s) took enormously more time to check their satisfiability than for the rest of the concepts. Since subsumption testing is reduced to satisfiability checking, it is at least *prima facie* plausible that the stand alone satisfiability time is correlated with a "hot spot". Indeed, the authors were able to "repair" their sample ontologies, by removing a small number of axioms based on guidance from SAT times.

5.1 Hot Spot Detection

Just knowing the "hard" concepts does not give us a corresponding set of axioms. For a candidate C, we use the $\top\bot^*$-module of the terms co-occurring with C in an axiom in $\mathcal{O}$ as the module "around" C. This roughly approximates what an ideal user might do: identify the problem (C) and then "remove it" (i.e., remove its explicit and implicit presence; the usage gets the explicit while the module gets the rest; this is an approximation, obviously). We rely on $\top\bot^*$-modules as these were shown to be the smallest kind of locality-based module [17]. The full technique is described by Algorithm 1. To test whether our indicator is effective, we compare it to candidates generated from randomly selected concepts. For each member of our set of 12 "hard" BioPortal ontologies we attempted to find 3 witness hot spots while testing no more than 1,000 hot spot candidates. In each case, we selected candidate hot spots using both the SAT-guided and the randomly selected concept methods.

Algorithm 1. Identification of hot spots in ontologies.

```
Input: Ontology $\mathcal{O}$
Output: Set of modules $S$, wherein for each $\mathcal{M}_i \in S$: $\mathrm{RT}(\mathcal{O} \setminus \mathcal{M}_i) \ll \mathrm{RT}(\mathcal{O})$

$S \leftarrow \emptyset$;  $Candidates \leftarrow \emptyset$;  $Times \leftarrow \emptyset$;  $max = 1000$;
for all atomic concepts $C \in \widetilde{\mathcal{O}}$ do {S1: Get SAT times}
    $Times \leftarrow Times \cup \langle C, SATtime(C) \rangle$
end for
Sort $Times$ in descending order of $SATtime(C)$
$Candidates \leftarrow Candidates \cup \{C$ with highest $SATtime$ up to $max$ concepts$\}$
for all $C \in Candidates$ do {S2: Verify candidate hot spots}
    $\mathcal{M} = \top\bot^*\text{-}mod(\{t \mid t$ co-occurs with $C$ in some $\alpha \in O\}, \mathcal{O})$
    if $\mathrm{RT}(\mathcal{O} \setminus \mathcal{M}) \ll \mathrm{RT}(\mathcal{O})$ then {S3: Test hot spot effectiveness}
        $S \leftarrow S \cup \mathcal{M}$
    end if
end for
return $S$
```

The first striking result is that we verified all the coarse-grained heterogeneity predictions. That is, if an ontology had a linear performance growth curve then neither method found a hot spot, whereas if the growth curve was non-linear then we found at least 1 hot spot, and usually 3.[10]

The hot spots found are described in Table 1. Both techniques were able to find hot spots most of the time, though the random approach failed in two cases. For the NEMO/HermiT combination, both approaches failed to find 3 before the limit, which suggests that hot spots are scarce. Contrariwise, for NCIt/HermiT, while the random approach failed to find any hot spots, the SAT-guided approach found them in 7 tests. In general, though not always, the SAT-guided approach found 3 hot spots in far fewer tests than the random approach (on average, respectively, in 129 vs. 426 tests), validating concept satisfiability as a significant indicator. Note that, at this point, we only present classification time boosts, the completeness of classification results is presented in Table 5.

A difficulty of the SAT-guided approach is the time to test all concepts for satisfiability. For example, we were unable to retrieve precise satisfiability-checking times for the GO-Ext ontology with FaCT++ and JFact. Instead, we used a timeout on each concept satisfiability check of 60 seconds. Also note that, for

[10] Of course, this could be just that we failed to find the telltale hot spots in the linear-growth ontologies. However, the overall evidence is highly suggestive.

Table 1. Comparison of hot spots found via SAT-guided (white rows) and random (grey rows) concept selection approach. "Nr. Tests" is the number of candidates tested before either finding 3 hot spots or exhausting the set search space (either the number of concepts in the ontology or 1000, whichever is smaller). CPU times in seconds.

Ontology	Nr. Axioms	Nr. Concepts	Reasoner	RT($\mathcal{O}$)	Hot Spots	Avg. RT($\mathcal{O} \setminus \mathcal{M}$)	Avg. Boost	Nr. Tests	Avg. $\lvert\mathcal{M}\rvert$	Avg. %$\lvert\mathcal{O}\rvert$	Avg. RT($\mathcal{M}$)
ChEBI	60,085	28,869	Pellet	65.8	3	12.3	82%	3	186	0.3%	0.55
					3	3.5	95%	89	522	1%	0.72
EFO	7,493	4,143	Pellet	61.1	3	9.6	81%	128	68	1%	0.13
					3	10.9	82%	863	70	1%	0.14
GO-Ext.	60,293	30,282	Pellet	268.4	3	29.6	89%	36	98	0.2%	0.08
					3	31.9	88%	419	17	0.03%	0.06
IMGT	1,112	112	Pellet	>54,000	1	26.1	99%	112	98	9%	0.09
					1	26.1	99%	112	98	9%	0.09
			HermiT	80.4	3	7.8	90%	86	35	3%	8.86
					3	7.1	91%	103	36	3%	10.4
NEMO	2,405	1,422	HermiT	76.3	1	5.5	93%	1,000	44	2%	4.63
					0	-	-	1,000	-	-	-
OBI	25,257	3,060	HermiT	61.6	3	2.3	96%	3	570	2%	1.56
					3	4.3	93%	189	576	2%	1.48
			JFact	72.1	3	1.1	93%	3	570	2%	1.12
					3	7.4	90%	57	576	2%	1.19
			Pellet	119.8	3	11.1	91%	29	708	3%	2.05
					3	21.6	82%	133	593	2%	1.76
VO	8,488	3,530	Pellet	4275.9	3	30.4	99%	11	322	4%	1.56
					3	371.7	91%	725	262	3%	0.61
NCIt	116,587	83,722	HermiT	430.1	3	16.1	88%	7	3,611	3%	16.14
					0	-	-	1,000	-	-	-

this particular ontology, we use (in Table 1 and subsequent ones) the median time value from the wide range of obtained classification times.

Overall, the hot spot finding mechanism described in Algorithm 1 is feasible, and successfully identified hot spots in all ontologies deemed performance heterogenous. Its run time is faster than the original classification time in 4 out of 11 cases, including one case (IMGT/Pellet) for which classification did not terminate within 15 hours. In general, the found hot spots were quite good: they typically were smaller than our limit (only IMGT/Pellet was above 5% of the ontology) and often giving massive speedups (e.g., IMGT/Pellet). There is no indication that hot spots, on their own, are particularly hard, which suggests an interaction effect, as expected.

5.2 Hot Spot Analysis

In order to investigate whether the removal of each hot spot happened to shift expensive constructs from the main input to the subset, we verify the expressivity of the hot spots and the remainder ontology (shown in Table 2).

Notice that, in several cases, the removal of the hot spot does not change the expressivity of the remainder w.r.t. the whole ontology, e.g. in ChEBI. However in other, yet few cases, there is a reduction of expressivity, e.g., the hot spots

Table 2. Expressivity of each original ontology ($\mathcal{O}$), its various hot spots ($\mathcal{M}$) and corresponding remainders ($\mathcal{O} \setminus \mathcal{M}$)

Ontology	$\mathcal{O}$	$\mathcal{O} \setminus \mathcal{M}$	$\mathcal{M}$
ChEBI	$\mathcal{ALE}+$	$\mathcal{ALE}+$	$\mathcal{ALE}+$
EFO	$\mathcal{SHOIF}$	$\mathcal{SHIF}$	$\mathcal{SHOIF}$
GO-Ext.	$\mathcal{ALEH}+$	$\mathcal{ALEH}+$	$\mathcal{AL}$, $\mathcal{ALEH}+$, $\mathcal{ALE}$
IMGT	$\mathcal{ALCIN}$	$\mathcal{ALC}$, $\mathcal{ALCIN}$	$\mathcal{ALCI}$, $\mathcal{ALCIN}$
NEMO	$\mathcal{SHIQ}$	$\mathcal{SHIF}$	$\mathcal{SHIQ}$
OBI	$\mathcal{SHOIN}$	$\mathcal{SHOIN}$	$\mathcal{SHOIF}$, $\mathcal{SHOIN}$
VO	$\mathcal{SHOIN}$	$\mathcal{SHOIN}$	$\mathcal{SHOIF}$
NCIt	$\mathcal{SH}$	$\mathcal{ALCH}$	$\mathcal{S}$

found in EFO leave the remainder without nominals. Similarly in NEMO the remainder no longer has qualified cardinality restrictions.

In order to get a better understanding of why this performance boost occurs, particularly the interaction effect between each hot spot and the ontology, we verify whether the removal of the hot spots from these ontologies changes the number of General Concept Inclusions (GCIs),[11] as these are an obvious potential source of hardness. The results gathered are shown in Table 3.

Table 3. Number of GCIs contained in the each ontology, its hot spots, and their corresponding remainders

Ontology	$\mathcal{O}$	$\mathcal{O} \setminus \mathcal{M}_1$	$\mathcal{O} \setminus \mathcal{M}_2$	$\mathcal{O} \setminus \mathcal{M}_3$	$\mathcal{M}_1$	$\mathcal{M}_2$	$\mathcal{M}_3$
EFO	172	163	164	164	9	8	8
GO-Ext	4407	4398	4382	4382	9	25	16
NCIt	42	37	36	36	5	6	6
NEMO	31	30	-	-	1	-	-
OBI	227	182	193	193	44	33	33
VO	235	196	201	197	39	34	38
IMGT	38	0	0	0	38	38	38

The obvious thing to notice here is that the removal of each of the 3 hot spots found within IMGT (for HermiT) leaves the remainder with no GCIs at all. Other cases are not so obvious, indeed in, e.g., NEMO or NCIt, only a few GCIs are removed from the original ontology. However, there seems to be some relation between the loss of GCIs from the original ontology into the hot spot, and the improvement in classification time. We speculate that a glass box approach to investigating this relation may help disentangle performance difficulties in specific reasoners, though this is beyond the scope of the paper.

[11] Axioms with complex concepts on both sides, e.g., $\exists r.A \sqsubseteq \exists r.B$.

5.3 Comparison with Pellint

As a final check, we compared our technique with Pellint [12]. Pellint is a "performance lint" dedicated specifically to the Pellet reasoner; it draws on the knowledge of the Pellet developers to generate a set of rules for what sorts of constructs and modelling patterns are likely to cause performance degradation when using Pellet — essentially it is a Pellet specific, ontology performance tuning expert system. Pellint not only identifies problem constructs, but it suggests approximations (typically by weakening or rewriting axioms) which "should" improve performance. If the number of axioms touched by Pellint repairs is sufficiently small and the gain sufficiently large, then Pellint will have identified a hot spot (though, at most 1). Since we believe that the "predicted homogeneous" ontologies have no hot spots (and we did not find any), we would expect that, while perhaps improving their performance, Pellint would not identify a hot spot. Similarly, for non-Pellet reasoners, we would expect no improvements at all. To check these conjectures we ran Pellint on all our ontologies and compared reasoning times for all "bad" reasoner/ontology combinations for both the Pellint approximated versions, and by removing the modified axioms (thus providing a direct comparison with Table 1). The results are shown in Table 4. Note that ontologies for which Pellint found no lints at all are omitted (5, in total). If Pellint found lints but could not alter them, then the number of altered axioms will read as 0 and no tests performed.

Table 4. Ontologies for which Pellint found "lints"

Ontology	Reasoner	RT($\mathcal{O}$)	Nr. Axioms Altered (lints)	%$\|\mathcal{O}\|$ Altered	Altered($\mathcal{O}$)		$\mathcal{O} \setminus$ {lints}	
					RT($\mathcal{O}$)	Boost	RT($\mathcal{O}$)	Boost
ChEBI	Pellet	65.8	0	-	-	-	-	-
EFO	Pellet	61.1	172	2%	3.7	94%	3.1	95%
GO-Ext.	Pellet	268.4	4407	7%	19.4	93%	5.85	98%
VO	Pellet	4275.9	231	3%	119.7	97%	3.32	99%
NCIt	HermiT	430.1	42	0.04%	443.4	-3%	448.1	-4%
Coriell	Pellet	923.5	46	0.03%	642.3	30%	631.0	32%
	FaCT++	156.1			159.2	-2%	159.1	-2%
	JFact	154.8			154.2	0.4%	143.9	7%
PRPPO	Pellet	118.9	0	-	-	-	-	-

First, Pellint was not able to find any hot spots in the predicted homogeneous ontologies, though for one (Coriell/Pellet) it was able to provide a significant performance boost (32%). This further confirms our linear/homogeneous hypothesis. Second, Pellint found hot spots in 3 out of 8 heterogeneous ontologies, performing much worse than even random concept selection. When found, the hot spots where competitive, but not all repaired lints improved performance (i.e., NCIt/HermiT). Pellint failed to find hot spots in our experiments due to finding no lints (5 ontologies), having no repairs[12] (2 ontologies), or just failing

[12] The set of suspect axioms might be a hot spot (or a subset thereof), but without access to them we cannot test.

to produce a dramatic enough (or any) effect (4 ontology/reasoner pairs, with most being non-Pellet). As expected, Pellint found no hot spots or performance improvements for other reasoners. Of course, this might be just be due to its overall poor hot spot finding.

Finally, Pellint's alterations had a noticeable negative effect on reasoning time compared to simple removal. Whether these approximations significantly save entailments needs to be investigated. Given the high development and maintenance costs of Pellint, it does not seem viable compared to search based methods.

6 Improving Classification via Hot Spots

The applicability of our hot spot finding method is dependent on how much information users are willing to lose. In a realistic edit-compile-deploy scenario, users may be wary to dispose of parts of their ontology. Thus, in order to avoid this predicament, we explore a series of approximation and knowledge compilation techniques, and compare them with a known approximate reasoning method. The latter is based on a reduction of the input into the tractable fragment of OWL: $\mathcal{EL}$, as implemented in TrOWL [15]. We implemented the $\mathcal{EL}$ reduction algorithm so as to apply it to any given reasoner other than REL (the reasoner used within TrOWL). Our approximation-based classifier is denoted *ELC*.

6.1 Approximate Reasoning

First off, given a hot spot and an ontology, we have an immediate approximation $\mathcal{O} \setminus \mathcal{M}$ of $\mathcal{O}$; It is much easier to reason over than the original ontology, though possibly too incomplete w.r.t. $Cl(\mathcal{O})$ (i.e., the set of inferred atomic subsumptions of $\mathcal{O}$). From hereon we derived two more approximations: 1) $Cl(\mathcal{O} \setminus \mathcal{M}) \cup Cl(\mathcal{M})$, which would naturally be more complete than $\mathcal{O} \setminus \mathcal{M}$ alone, and 2) $\mathcal{O} \setminus \mathcal{M} \cup Cl(\mathcal{M})$, where we expect that the interaction between inferred subsumptions in $\mathcal{M}$ and the remainder will bring us closer to $Cl(\mathcal{O})$. A comparison of these techniques is shown in Table 5, containing the results of each of the 3 approximations as well as *ELC* with the respective reasoner.

Overall the closest approximation is $\mathcal{O} \setminus \mathcal{M} \cup Cl(M)$, which yields an average completeness of 99.84% and an average boost of 89.3% over the original times. *ELC* is typically more complete, though in several cases classifying an ontology with *ELC* is much slower than the original RT($\mathcal{O}$), e.g., *ELC* failed to classify the NCIt within 5 hours, compared to ≈7 minutes originally. Similarly with ChEBI and OBI, the approximation is no faster than the original times. Overall the average boost via *ELC* is non-existent, particularly due to the NCIt case. By excluding that one case, *ELC*'s average classification time boost is of 33.7%.

Applying the original TrOWL system, with its internal reasoner REL, is not so much better than using standard DL reasoners on the $\mathcal{EL}$ approximations, particularly since some DL reasoners (e.g., Pellet or FaCT++) are finely tuned to the $\mathcal{EL}$ fragment of OWL. Nevertheless, we analysed those results only to find that TrOWL has the exact same problem with the NCIt, and out-performs *ELC* in 4 out of 7 cases by mere seconds.

Table 5. Approximate reasoning results for the approximations $\mathcal{O} \setminus \mathcal{M}$, $Cl(\mathcal{O} \setminus \mathcal{M}) \cup Cl(\mathcal{M})$, $\mathcal{O} \setminus \mathcal{M} \cup Cl(M)$, and, finally, ELC. The completeness of each approach w.r.t. $Cl(\mathcal{O})$ is denoted "Compl.".

Ontology	Reasoner	$\mathcal{O} \setminus \mathcal{M}$		$Cl(\mathcal{O} \setminus \mathcal{M}) \cup Cl(\mathcal{M})$		$\mathcal{O} \setminus \mathcal{M} \cup Cl(M)$		ELC	
		Compl.	Boost	Compl.	Boost	Compl.	Boost	Compl.	Boost
ChEBI	Pellet	55%	89%	55%	89%	100%	84%	100%	-207%
EFO	Pellet	78%	86%	79%	86%	100%	81%	100%	63%
NCIt	HermiT	75%	90%	80%	87%	100%	89%	-[12]	-2651%
NEMO	HermiT	97%	96%	98%	92%	100%	96%	99.94%	93%
OBI	HermiT	51%	96%	55%	94%	100%	94%	100%	14%
	JFact	51%	91%	55%	90%	99.92%	84%	99.95%	-10%
	Pellet	50%	88%	54%	88%	100%	86%	100%	54%
IMGT	Pellet	68%	100%	76%	100%	100%	100%	100%	100%
	HermiT	92%	92%	97%	78%	99.92%	92%	100%	100%
VO	Pellet	50%	98%	52%	98%	98.36%	94%	100%	97%
GO-Ext	Pellet	95%	90%	96%	90%	100%	81%	100%	33%
Average		69.2%	92.4%	72.3%	90.4%	99.84%	89.3%	99.99%	-210%

6.2 Knowledge Compilation

While the loss of entailments via our best approximation is typically empty, or very low, we investigate whether a number of knowledge compilation techniques based on hot spots enjoy the same performance boosts as the approximations in Section 6.1. These techniques all maintain 100% completeness of knowledge contained in the original ontologies, i.e., they produce logically equivalent knowledge bases. The rationale behind these techniques is that by adding inferred knowledge (e.g., from a hot spot) to the original ontology, reasoners will not need to do certain (possibly expensive) subsumption tests, and, as a consequence, should (at least intuitively) perform faster. The results are shown in Table 6.

First thing to notice here is that adding the inferred class hierarchy of the parts does not necessarily improve classification time over the whole. There are cases, such as OBI with JFact, where all compilation techniques took much longer to classify than the original (note that we timed-out the operation at 5 hours). On the other hand, there are cases where there is mild to noteworthy improvement, for instance VO classifies 75% faster when we use the second compilation technique, which is a significant improvement with no loss of knowledge. Similarly the GO-Ext ontology classifies 92% faster with both the second and third compilation technique. Nevertheless, the results gathered are not nearly as stable w.r.t. classification time improvement as our approximations, and the improvements obtained are also not as high as those shown in Section 6.1.

[12] The classification of the NCIt was interrupted after running for 5 hours, well above the original classification time.

Table 6. Compilation results for the techniques $\mathcal{O} \cup Cl(\mathcal{M})$, $\mathcal{O} \cup Cl(\mathcal{O} \setminus \mathcal{M})$ and $\mathcal{O} \cup Cl(\mathcal{M}) \cup Cl(\mathcal{O} \setminus \mathcal{M})$

Ontology	Reasoner	$\mathcal{O} \cup Cl(\mathcal{M})$		$\mathcal{O} \cup Cl(\mathcal{O} \setminus \mathcal{M})$		$\mathcal{O} \cup Cl(\mathcal{M}) \cup Cl(\mathcal{O} \setminus \mathcal{M})$	
		Time	Boost	Time	Boost	Time	Boost
ChEBI	Pellet	74.5	18%	73.1	19%	73.6	19%
EFO	Pellet	51.3	30%	63	14%	62.9	14%
NCIt	HermiT	616.1	6%	603.2	8%	614.5	6%
NEMO	HermiT	94.9	5%	94.6	6%	98.6	2%
OBI	HermiT	71	-3%	69.1	0%	70.7	-2%
	JFact	>5hrs	-	>5hrs	-	>5hrs	-
	Pellet	264	-66%	207.5	-31%	276.6	-74%
IMGT	Pellet	36000	33%	36000	33%	36000	33%
	HermiT	94.8	-4%	94.9	-4%	94.8	-4%
VO	Pellet	1704.4	60%	1066.2	75%	2136.2	50%
GO-Ext	Pellet	161.4	56%	30.1	92%	30.6	92%
Average Boost		-	14%	-	21%	-	14%

7 Related Work

In [23], a number of ontology profiling techniques are proposed and realized in a tool, Tweezers for Pellet. Tweezers allows users to investigate performance statistics, such as the satisfiability checking time for each concept in the ontology, but relies on the user to apply this information. Our goal driven technique can be seen as the automated exploitation of their statistics.

In [4] the author proposes three techniques to automatically identify potentially expensive "constructs" (concepts, roles or axioms). These techniques search for "suspect" constructs by recursively splitting an ontology in different manners, and individually testing performance over the parts until suspects are found. While their actual attempt was rather *ad hoc*, it does suggest an alternative discovery mechanism (as they did find some hot spots).

In [3] the authors present a form of OWL reasoner benchmarking based on justifications. JustBench computes all justifications for entailments in a given ontology, and measures the performance of reasoners on those justifications. The authors hoped that they would find justifications that were hot spots themselves (or indicators thereof), but this hope was not borne out by their experiments.

8 Discussion and Applications

Unlike with hot spots in programs, there is no straightforward relationship between the performance of a "hot spot" in isolation and the effect it has on the ontology as a whole (see the last column in Table 1). Our results have shown that there is no precise co-relation between the classification time of a hot spot alone, and the reduction in classification time when such hot spot is removed. This is somewhat similar to the fact that in a program, if an individually quick function is called sufficiently often, it may be the performance bottleneck for that program. That is, looking at the performance of the function in isolation is not sufficient to determine its effect on the overall runtime. However, in our case, there are many possible and currently unknown ways that a performance hot spot might affect overall runtime, and yet not exhibit pathological behaviour on its own. Indeed, the fact that sometimes adding axioms is sufficient to eliminate performance problems shows that isolating behaviour is not a reliable predictor of integrated effect. It would be interesting to seek out inverse hot spots, that is, acceptably small subsets whose removal greatly *increases* the classification time of an ontology, though these would have less end user applicability. Of course, merely finding hot spots does not provide any explanation of performance patterns, it merely provides tools for investigating them. On the other hand, it is a purely black box technique, thus, unlike Pellint, does not require such insight to be effective.

Our investigation was partly inspired by our observation of user coping techniques for recalcitrant ontologies, thus it is natural to seek to apply them in such scenarios. The basic idea is straightforward enough: Present the user with a selection of hot spots and let them select the most appropriate one to "set aside" (permanently or temporarily) or to rewrite into a less damaging approximation. Of course, we might want hot spots with somewhat different properties, e.g., that the remainder ontology is a module rather than the hot spot, so that "safe edits" to the remainder will not alter the meaning of the hot spot. We might use heuristics to select a hot spot for automated removal or approximation. Modular hot spots might be presented to the user so they can attempt to have a clearer understanding of "what was removed."

Our techniques could benefit reasoner developers as well. For example, a hot spot gives the developer a pair of almost identical ontologies with vastly different performance behaviour. By comparing the profiling reports on their reasoners processing these inputs, the developer might gain additional insight.

Currently, we have concentrated on satisfiability-checking time of atomic concepts as the indicator for hot spots. There are clearly alternatives for this, e.g., small atoms [6] or justifications, as well as brute force methods [4].

All our experiments, as they stand, can be improved in two dimensions: 1) more input ontologies are always better, and 2) our sampling, particularly in the coarse grained method, is very low. Clearly, they were sufficient to reveal some interesting phenomena, but not to establish statistically significant findings.

Finally, it may be possible to derive Pellint-like rules directly from hot spots extracted from a large number of ontologies. While requiring maintenance, it would be inherently much faster than our approaches as it would not require any reasoning at all.

References

1. Baader, F., Lutz, C., Suntisrivaraporn, B.: CEL — A Polynomial-Time Reasoner for Life Science Ontologies. In: Furbach, U., Shankar, N. (eds.) IJCAR 2006. LNCS (LNAI), vol. 4130, pp. 287–291. Springer, Heidelberg (2006)

2. Baader, F., Lutz, C., Suntisrivaraporn, B.: Efficient reasoning in $\mathcal{EL}^+$. In: Proc. of DL 2006 (2006)
3. Bail, S., Parsia, B., Sattler, U.: JustBench: A Framework for OWL Benchmarking. 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. 32–47. Springer, Heidelberg (2010)
4. Charaniya, S.: Facilitating DL Reasoners Through Ontology Partitioning. Master's thesis, Nagpur University, India (2006)
5. Cuenca Grau, B., Horrocks, I., Kazakov, Y., Sattler, U.: Modular reuse of ontologies: Theory and practice. J. of Artificial Intelligence Research 31 (2008)
6. Del Vescovo, C., Parsia, B., Sattler, U., Schneider, T.: The modular structure of an ontology: Atomic decomposition. In: Proc. of IJCAI 2011 (2011)
7. Horridge, M., Bechhofer, S.: The OWL API: A Java API for working with OWL 2 ontologies. In: Proc. of OWLED 2009 (2009)
8. Horrocks, I., Kutz, O., Sattler, U.: The even more irresistible $\mathcal{SROIQ}$. In: Proc. of KR 2006 (2006)
9. Horrocks, I.: The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press (2003)
10. Horrocks, I., Patel-Schneider, P.F.: Evaluating optimised decision procedures for propositional modal k(m) satisfiability. J. of Automated Reasoning 28, 173–204 (2002)
11. Kalyanpur, A., Parsia, B., Horridge, M., Sirin, E.: Finding All Justifications of OWL DL Entailments. 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.) ISWC/ASWC 2007. LNCS, vol. 4825, pp. 267–280. Springer, Heidelberg (2007)
12. Lin, H., Sirin, E.: Pellint - a performance lint tool for Pellet. In: Proc. of OWLED-08EU (2008)
13. 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, W170–W173 (2009)
14. Patel-Schneider, P.F., Sebastiani, R.: A new general method to generate random modal formulae for testing decision procedures. J. of Artificial Intelligence Research 18, 351–389 (2003)
15. Ren, Y., Pan, J.Z., Zhao, Y.: Soundness Preserving Approximation for TBox Reasoning. In: Proc. of AAAI 2010 (2010)
16. Rudolph, S., Tserendorj, T., Hitzler, P.: What Is Approximate Reasoning? In: Calvanese, D., Lausen, G. (eds.) RR 2008. LNCS, vol. 5341, pp. 150–164. Springer, Heidelberg (2008)
17. Sattler, U., Schneider, T., Zakharyaschev, M.: Which kind of module should I extract? In: Proc. of DL 2009 (2009)
18. Schaerf, M., Cadoli, M.: Tractable reasoning via approximation. Artificial Intelligence 74, 249–310 (1995)
19. Shearer, R., Motik, B., Horrocks, I.: HermiT: A highly-efficient OWL reasoner. In: Proc. of OWLED-08EU (2008)
20. Sirin, E., Parsia, B., Cuenca Grau, B., Kalyanpur, A., Katz, Y.: Pellet: A practical OWL-DL reasoner. J. of Web Semantics 5(2) (2007)

21. Tsarkov, D., Horrocks, I.: FaCT++ Description Logic Reasoner: System Description. In: Furbach, U., Shankar, N. (eds.) IJCAR 2006. LNCS (LNAI), vol. 4130, pp. 292–297. Springer, Heidelberg (2006)
22. W3C OWL Working Group: OWL 2 Web Ontology Language: Document overview. W3C Recommendation (October 27, 2009), http://www.w3.org/TR/owl2-syntax/
23. Wang, T.D., Parsia, B.: Ontology Performance Profiling and Model Examination: First Steps. 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.) ISWC/ASWC 2007. LNCS, vol. 4825, pp. 595–608. Springer, Heidelberg (2007)

Concept-Based Semantic Difference in Expressive Description Logics

Rafael S. Gonçalves, Bijan Parsia, and Ulrike Sattler

School of Computer Science
University of Manchester
Manchester, United Kingdom

Abstract. Detecting, much less understanding, the difference between two description logic based ontologies is challenging for ontology engineers due, in part, to the possibility of complex, non-local logic effects of axiom changes. First, it is often quite difficult to even determine which concepts have had their meaning altered by a change. Second, once a concept change is pinpointed, the problem of distinguishing whether the concept is directly or indirectly affected by a change has yet to be tackled. To address the first issue, various principled notions of "semantic diff" (based on deductive inseparability) have been proposed in the literature and shown to be computationally practical for the expressively restricted case of $\mathcal{ELH}^r$-terminologies. However, problems arise even for such limited logics as $\mathcal{ALC}$: First, computation gets more difficult, becoming undecidable for logics such as $\mathcal{SROIQ}$ which underly the Web Ontology Language (OWL). Second, the presence of negation and disjunction make the standard semantic difference too sensitive to change: essentially, any logically effectual change always affects all terms in the ontology. In order to tackle these issues, we formulate the central notion of finding the *minimal change set* based on model inseparability, and present a method to differentiate changes which are specific to (thus directly affect) particular concept names. Subsequently we devise a series of computable approximations, and compare the variously approximated change sets over a series of versions of the NCI Thesaurus (NCIt).

1 Introduction

Determining the significant differences between two documents (so-called "diff") is a standard and significant problem across a wide range of activities, notably software development. Standard textual diffing algorithms perform poorly on description logic (DL) based ontologies, both for structural reasons (e.g., ontology serializations, such as those of OWL, tend not to impose stable ordering of axioms), and due to the highly non-local and unintuitive logical effects of changes to axioms. This gave rise to several diff notions for OWL ontologies, encompassing various types of change detection and impact analysis mechanisms. Within change detection there are two key dimensions of change: *syntactic* and *semantic*, leading to syntactic and semantic diffs. The former, e.g. those diffs based on OWL's notion of "structural equivalence" [3,7,11], detect asserted changes between ontologies, and thus are of interest for, e.g., versioning. In [3] we addressed the problem of identifying and characterising the impact of such asserted changes by, for instance, pinpointing whether each change produces a logical effect. However, that work focused exclusively on axiom level analysis. Since OWL ontologies are sets of axioms, this is a natural level of analysis. However, ontologies often serve as ways to manage controlled vocabularies, that is, the set of axioms is a development time artifact supporting the delivery of a hierarchically organized set of categorical *terms*. In such cases, end users are most directly concerned with changes to the terms themselves and may not even have access to the axioms. Thus, the modeller must not only be aware of the axioms they have touched, but how those changes affect the concepts in the ontology.

For the purpose of determining entailment (and ergo term) differences, recent notions of semantic difference based on conservative extensions have provided a robust theoretical and practical basis for analysing these logical effects [6]. Unfortunately, semantic difference is computationally expensive even for inexpressive logics such as $\mathcal{EL}$. For the very expressive logics such as $\mathcal{SROIQ}$ (the DL underlying OWL 2) it is undecidable [9]. Furthermore, standard semantic difference runs into other difficulties in more expressive logics when we consider differences w.r.t. all terms in both ontologies. In particular, if we compare entailment sets over logics with disjunction and negation we end up with vacuously altered terms: any logically effectual change will alter the meaning of every term.

To address this vacuity problem, we present *a non-trivializable notion of semantic difference of concepts*, which includes *a mechanism for distinguishing directly and indirectly affected concepts*. To address the undecidability of even our refined semantic difference problem for the $\mathcal{SROIQ}$ (i.e., OWL 2) DL, we define *a series of motivated semantic diff approximations for expressive description logics*. These algorithms are evaluated on a select subset of the National Cancer Institute (NCI) Thesaurus (NCIt) corpus, by a comparison of the changes found via the proposed approximations and related approaches. Our experiments show that our strongest approximation, "Grammar diff", finds significantly more changes than all other methods across the corpus, and far more than are identified in the NCIt change logs. We show that distinguishing direct and indirect changes is necessary for making concept based change logs manageable.

2 Preliminaries

We assume the reader to be familiar with ontologies and OWL, as well as the underlying description logics (DLs) [1]. We use *terms* to refer to concept and role names. When comparing two ontologies we refer to them as $\mathcal{O}_1$ and $\mathcal{O}_2$, and their *signatures* (i.e., the set of terms occurring in them) as $\widetilde{\mathcal{O}}_1$ and $\widetilde{\mathcal{O}}_2$, respectively. Throughout this paper we refer to $\widetilde{\mathcal{O}}_1 \cup \widetilde{\mathcal{O}}_2$ as Σ_u. This signature is the natural subject of comparison of terminological changes between two ontologies. The signature of an axiom α is denoted $\widetilde{\alpha}$.

Throughout this paper we use the standard description and first order logic notion of entailment; an axiom α entailed by an ontology $\mathcal{O}$ is denoted $\mathcal{O} \models \alpha$. We refer to an *effectual* addition (removal) from $\mathcal{O}_1$ to $\mathcal{O}_2$ as an axiom α such that $\alpha \in \mathcal{O}_2$ and $\mathcal{O}_1 \not\models \alpha$ ($\alpha \in \mathcal{O}_1$ and $\mathcal{O}_2 \not\models \alpha$) [3]. Thus two ontologies are logically equivalent, denoted $\mathcal{O}_1 \equiv \mathcal{O}_2$, if there is no effectual change (addition or removal) between $\mathcal{O}_1$ and $\mathcal{O}_2$. The set of *subconcepts* of an ontology $\mathcal{O}$ is recursively defined as all subconcepts found in each axiom of $\mathcal{O}$, plus $\{\top, \bot\}$. W.l.o.g, we define all diff functions asymmetrically, thus to get the full diff between two ontologies we compute $\text{Diff}(\mathcal{O}_1, \mathcal{O}_2)$ (for additions) and $\text{Diff}(\mathcal{O}_2, \mathcal{O}_1)$ (for removals).

The restriction of an interpretation $\mathcal{I}$ to a set of terms Σ is denoted $\mathcal{I}|_\Sigma$. Two interpretations $\mathcal{I}$ and $\mathcal{J}$ coincide on a signature Σ (denoted $\mathcal{I}|_\Sigma = \mathcal{J}|_\Sigma$) if $\Delta^\mathcal{I} = \Delta^\mathcal{J}$ and $t^\mathcal{I} = t^\mathcal{J}$ for each $t \in \Sigma$.

Throughout this paper we use the notion of model conservative extension (mCE) [2,9], and associated inseparability relation [13]. The axioms expressible in a DL $\mathcal{L}$ over a set of terms Σ is denoted $\mathcal{L}(\Sigma)$.

Definition 1. *Given two ontologies $\mathcal{O}_1$, $\mathcal{O}_2$ over a DL $\mathcal{L}$, and a signature Σ.*

(1) $\mathcal{O}_2$ *is model Σ-inseparable from* $\mathcal{O}_1$ ($\mathcal{O}_1 \equiv^{mCE}_\Sigma \mathcal{O}_2$) *if*

$$\{\mathcal{I}|_\Sigma \mid \mathcal{I} \models \mathcal{O}_1\} = \{\mathcal{J}|_\Sigma \mid \mathcal{J} \models \mathcal{O}_2\}$$

(2) $\mathcal{O}_2$ *is deductive Σ-inseparable from* $\mathcal{O}_1$ *w.r.t.* $\mathcal{L}$ ($\mathcal{O}_1 \equiv^{\mathcal{L}}_\Sigma \mathcal{O}_2$) *if*

$$\{\alpha \in \mathcal{L}(\Sigma) \mid \mathcal{O}_1 \models \alpha\} = \{\alpha \in \mathcal{L}(\Sigma) \mid \mathcal{O}_2 \models \alpha\}$$

(3) $$\text{Diff}(\mathcal{O}_1, \mathcal{O}_2)^{\mathcal{L}}_\Sigma = \{\eta \in \mathcal{L}(\Sigma) \mid \mathcal{O}_1 \not\models \eta \textit{ and } \mathcal{O}_2 \models \eta\}$$

Note that $\text{Diff}(\mathcal{O}_1, \mathcal{O}_2)^{\mathcal{L}}_\Sigma = \emptyset$ if and only if $\mathcal{O}_1 \equiv^{\mathcal{L}}_\Sigma \mathcal{O}_2$. Also, bear in mind (esp. for the running example) that $\mathcal{O}_1 \not\equiv^{\mathcal{L}}_\Sigma \mathcal{O}_2$ implies $\mathcal{O}_1 \not\equiv^{mCE}_\Sigma \mathcal{O}_2$. In the remainder of this paper we use $\mathcal{SROIQ}$ General Concept Inclusions (GCIs) for $\mathcal{L}$, and omit $\mathcal{L}$ if this is clear from the context.

3 Exisiting Semantic Diff

$\text{Diff}(\mathcal{O}_1, \mathcal{O}_2)_\Sigma$ alone, if non-empty, tells us that there are new entailments expressed in the designated signature, but does not pick out specific terms in that signature. The CEX [6] diff method focuses on elements of $\text{Diff}(\mathcal{O}_1, \mathcal{O}_2)_\Sigma$, so called *witness axioms*, with specific forms — subsumptions with an atomic left hand (resp. right hand) side, i.e., of the form $A \sqsubseteq C$ (resp. $C \sqsubseteq A$) where A is atomic and C is a possibly complex concept, called the *witness concept*. All terms that appear in those positions in axioms in $\text{Diff}(\mathcal{O}_1, \mathcal{O}_2)_\Sigma$ form the set of affected terms (denoted $\text{AT}(\mathcal{O}_1, \mathcal{O}_2)_\Sigma$). By restricting attention to changes to individual terms (rather than to sets of terms together), CEX 1) becomes decidable[1], 2) produces manageable diff reports ($\text{AT}(\mathcal{O}_1, \mathcal{O}_2)_\Sigma$ is a subset of the signature, not of the powerset of the signature), and 3) the diff report is nicely interpretable. CEX gets interpretability both by focusing on changes to individual terms in themselves (instead of on coordinated changes to sets of terms) and by exploiting the natural directionality of its witness axioms.

[1] At least for the restricted case of acyclic $\mathcal{ELH}^r$ terminologies ($\mathcal{EL}$ extended with role inclusions and range restrictions).

CEX divides $\mathrm{AT}(\mathcal{O}_1, \mathcal{O}_2)_\Sigma$ into specialised, denoted $\mathrm{AT}(\mathcal{O}_1, \mathcal{O}_2)^{\mathrm{L}}_{\Sigma}$, and generalised, designated $\mathrm{AT}(\mathcal{O}_1, \mathcal{O}_2)^{\mathrm{R}}_{\Sigma}$, concept names depending on whether a term appears on the left or right hand side of a witness axiom (the same term may appear in both). The CEX algorithm is sound and complete: If there is a witness axiom for a term A in the deductive closure of $\mathcal{O}_2$, then CEX will find it.

The computational complexity of deciding Σ-entailment is undecidable for expressive DLs such as $\mathcal{SROIQ}$. For $\mathcal{EL}$ it is already ExpTime-complete [10], while for $\mathcal{ALC}$, $\mathcal{ALCQ}$, and $\mathcal{ALCQI}$ it is 2ExpTime-complete [9]. Thus, CEX is computationally infeasible for expressive logics. Moreover, when considering differences over Σ_u, a direct extension of Σ-difference for more expressive logics such as $\mathcal{ALC}$ would be futile; when we step beyond $\mathcal{EL}$ as a witness language into more expressive logics with disjunction and negation, if $\mathcal{O}_1 \not\equiv \mathcal{O}_2$ then $\mathrm{AT}(\mathcal{O}_1, \mathcal{O}_2)_\Sigma$ contain all terms in Σ. Consider the following $\mathcal{EL}$ ontologies: $\mathcal{O}_1 = \{A \sqsubseteq B\}$, and $\mathcal{O}_2 = \{A \sqsubseteq B, C \sqsubseteq D\}$. Clearly $\mathcal{O}_2$ is a conservative extension of $\mathcal{O}_1$ w.r.t. $\Sigma = \{A, B\}$, but if we consider Σ_u then that is no longer the case; a witness axiom for the separability would be, e.g., $\eta := A \sqsubseteq \neg C \sqcup D$. This witness "witnesses" a change to every concept $A' \in \Sigma_u$; for each witness axiom $\eta' : A' \sqsubseteq \neg C \sqcup D$ we have that $\mathcal{O}_1 \not\models \eta'$, while $\mathcal{O}_2 \models \eta'$. Such a witness would suffice to pinpoint, according to Σ-difference, that all terms in Σ_u have changed: $\mathrm{AT}(\mathcal{O}_1, \mathcal{O}_2)_{\Sigma_u} = \Sigma_u$ since $\top \sqsubseteq \neg C \sqcup D$. Consequently, this kind of witness is uninteresting for any particular concept aside from $\top$. Likewise, a change $A \sqsubseteq \bot$ implies that, for all B in the signature of the ontology in question, we have that $A \sqsubseteq B$. Yet these consequences are of no interest to any concept B.

Similar to the case of the least common subsumer [8], the presence of disjunction (and negation) trivialises definitions that are meaningful in less expressive logics. Thus we need to refine our diff notion when dealing with propositionally closed witness languages.

A simple approach to coping with the dual problems of computational difficulty and triviality is to use a relatively inexpressive witness language. For example, ContentCVS [5] computes an approximation of $\mathrm{AT}(\mathcal{O}_1, \mathcal{O}_2)^{\mathrm{L}}_{\Sigma}$ (i.e., $A \sqsubseteq C$) for OWL 2 DL where C conforms to a specific grammar (B is atomic):

Grammar $G_{cvs} : C \longrightarrow B \mid \exists r.B \mid \forall r.B \mid \neg B$

Grammar G_{cvs} is a bit ad hoc, being based on the designers intuitions of what might be "interesting" but yet inherently finite. In a user study of ContentCVS, users criticised "the excessive amount of information displayed when using larger approximations of the deductive difference" [5]. The users were not presented with the affected terms directly, only via presentation of the witness axioms. ContentCVS avoids triviality by not allowing axiom encoding witness concepts.

4 Semantic Diff

Given the shortcomings of existing methodologies, and the triviality of Σ_u-difference in expressive ontologies, we present a semantic diff method that *a)* determines which concepts have been affected by changes. For exposition reasons, we concentrate on concepts, though roles are easily added. And *b)* identifies which concepts have been directly (or indirectly) changed. Ideally, a solution to these problems would be *1)* a computationally feasible diff function (for OWL 2 DL), *2)* based on a principled grammar, that *3)* returns those concept names affected by changes between two ontologies, while *4)* distinguishing whether each concept name is directly (or indirectly) specialised and/or generalised.

Consider the toy ontologies $\mathcal{O}_1$ and $\mathcal{O}_2$ defined in Table 1; they will be used throughout this section as a running example.

Table 1. Ontologies $\mathcal{O}_1$ and $\mathcal{O}_2$

$\mathcal{O}_1$		$\mathcal{O}_2$	
α_1 :	$A \sqsubseteq B$	β_1 :	$A \sqsubseteq B$
α_2 :	$B \sqsubseteq C$	β_2 :	$B \sqsubseteq C \sqcap D$
α_3 :	$D \sqsubseteq \exists r.E$	β_3 :	$D \sqsubseteq \exists r.E$
α_4 :	$E \sqsubseteq \forall s.G$	β_4 :	$E \sqsubseteq \forall s.(G \sqcap F)$
α_5 :	$\exists r.I \sqsubseteq J$	β_5 :	$\exists r.I \sqsubseteq J$
		β_6 :	$\forall t.H \sqsubseteq I$

4.1 Determining the Change Set

Given two ontologies $\mathcal{O}_1$ and $\mathcal{O}_2$, such that $\mathcal{O}_1 \not\equiv \mathcal{O}_2$ (i.e., there exists at least one effectual change in Diff($\mathcal{O}_1, \mathcal{O}_2$)), we know that $\mathcal{O}_1$ and $\mathcal{O}_2$ are not Σ-inseparable (for $\Sigma := \Sigma_u$) w.r.t. model inseparability, i.e., $\mathcal{O}_1 \not\equiv_{\Sigma}^{mCE} \mathcal{O}_2$ since an effectual change implies some change in semantics. In order to pinpoint this change, we need to find the set of terms Σ' s.t. $\mathcal{O}_1$ is mCE-inseparable from $\mathcal{O}_2$ w.r.t. the remaining signature $\Sigma \setminus \Sigma'$: $\mathcal{O}_1 \equiv_{\Sigma \setminus \Sigma'}^{mCE} \mathcal{O}_2$. Then we know that, from $\mathcal{O}_1$ to $\mathcal{O}_2$, there are no changes in entailments over $\Sigma \setminus \Sigma'$. We refer to this set of terms Σ' as the Minimal Change Set (denoted MinCS($\mathcal{O}_1, \mathcal{O}_2$)), in the sense that we can formulate a non-trivial witness axiom η over Σ' s.t. $\mathcal{O}_1 \not\models \eta$ but $\mathcal{O}_2 \models \eta$. Thus we denote these terms as *affected.*

Definition 2 (Minimal Affected Terms). *A set $\Sigma' \subseteq \Sigma$ is* a set of minimal affected terms *between $\mathcal{O}_1$ and $\mathcal{O}_2$ if:*

$$\mathcal{O}_1 \not\equiv_{\Sigma'}^{mCE} \mathcal{O}_2 \text{ and for all } \Sigma'' \subsetneq \Sigma' : \mathcal{O}_1 \equiv_{\Sigma''}^{mCE} \mathcal{O}_2.$$

The set of all such sets is denoted MinAT($\mathcal{O}_1, \mathcal{O}_2$)*.*

In order to form the minimal change set, we take the union over all sets of affected terms in MinAT($\mathcal{O}_1, \mathcal{O}_2$).

Definition 3 (Minimal Change Set). *The* minimal change set*, denoted* MinCS($\mathcal{O}_1, \mathcal{O}_2$)*, of two ontologies is defined as follows:*

$$\text{MinCS}(\mathcal{O}_1, \mathcal{O}_2) := \bigcup \text{MinAT}(\mathcal{O}_1, \mathcal{O}_2)\,, \textit{ and}$$
$$\text{MinCS}(\mathcal{O}_1, \mathcal{O}_2)^C := \{C \mid C \textit{ is a concept name in } \text{MinCS}(\mathcal{O}_1, \mathcal{O}_2)\}.$$

From the example ontologies in Table 1 we have that $\{A, D\}$ is a set of minimal affected terms between $\mathcal{O}_1$ and $\mathcal{O}_2$; $\mathcal{O}_1 \not\equiv_{\{A,D\}}^{mCE} \mathcal{O}_2$, because $\mathcal{O}_1 \not\models A \sqsubseteq D$

while $\mathcal{O}_2 \models A \sqsubseteq D$. $\{A, D\}$ is minimal since $\mathcal{O}_1 \equiv^{mCE}_{\{A\}} \mathcal{O}_2$, and similarly for $\{D\}$. Analogous cases can be made for $\{B, D\}$ via witness axiom $B \sqsubseteq D$, $\{E, s, F\}$ via $E \sqsubseteq \forall s.F$, $\{r, t, H, J\}$ via $\exists r.\forall t.H \sqsubseteq J$, and finally $\{t, H, I\}$ via β_6. So the minimal change set (restricted to concept names) between these two ontologies is $\text{MinCS}(\mathcal{O}_1, \mathcal{O}_2)^C := \{A, B, D, E, F, H, I, J\}$.

4.2 Characterising Concept Impact

Prior to determining how a concept in a signature Σ has changed (e.g., it has a new superconcept), we employ a diff function Φ which, given two ontologies and Σ, formulates *a set of witness axioms over* Σ, denoted $\Phi\,\text{Diff}(\mathcal{O}_1, \mathcal{O}_2)_\Sigma$, such that, for each $\eta \in \Phi\,\text{Diff}(\mathcal{O}_1, \mathcal{O}_2)_\Sigma$: $\mathcal{O}_1 \not\models \eta$ and $\mathcal{O}_2 \models \eta$. Now given such a set $\Phi\,\text{Diff}(\mathcal{O}_1, \mathcal{O}_2)_\Sigma$, we can tell apart specialised and generalised concepts depending on whether the witness concept is on the right or left hand side of the witness axiom, accordingly. Furthermore, we regard a concept name A as *directly specialised (generalised)* via some witness C if there is no concept name B that is a superconcept (subconcept) of A, and C is also a witness for a change in B. Otherwise A *changed indirectly.*

Definition 4. *A* diff function *Φ returns a subset $\Phi\,\text{Diff}(\mathcal{O}_1, \mathcal{O}_2)_\Sigma$ of* $\text{Diff}(\mathcal{O}_1, \mathcal{O}_2)_\Sigma$*. For a diff function Φ, the sets of* affected concept names *for a signature Σ are:*

$$\Phi\text{-}\,\text{AT}(\mathcal{O}_1, \mathcal{O}_2)^\top_\Sigma = \begin{cases} \{\top\} & \textit{if there is a } \top \sqsubseteq C \in \Phi\,\text{Diff}(\mathcal{O}_1, \mathcal{O}_2)_\Sigma \\ \emptyset & \textit{otherwise} \end{cases}$$

$$\Phi\text{-}\,\text{AT}(\mathcal{O}_1, \mathcal{O}_2)^\bot_\Sigma = \begin{cases} \{\bot\} & \textit{if there is a } C \sqsubseteq \bot \in \Phi\,\text{Diff}(\mathcal{O}_1, \mathcal{O}_2)_\Sigma \\ \emptyset & \textit{otherwise} \end{cases}$$

$$\Phi\text{-}\,\text{AT}(\mathcal{O}_1, \mathcal{O}_2)^\text{L}_\Sigma = \{A \in \Sigma \mid \textit{there exists } A \sqsubseteq C \in \Phi\,\text{Diff}(\mathcal{O}_1, \mathcal{O}_2)_\Sigma \textit{ and } \top \sqsubseteq C \notin \Phi\,\text{Diff}(\mathcal{O}_1, \mathcal{O}_2)_\Sigma\}$$

$$\Phi\,\text{AT}(\mathcal{O}_1, \mathcal{O}_2)^\text{R}_\Sigma = \{A \in \Sigma \mid \textit{there exists } C \sqsubseteq A \in \Phi\,\text{Diff}(\mathcal{O}_1, \mathcal{O}_2)_\Sigma \textit{ and } C \sqsubseteq \top \notin \Phi\,\text{Diff}(\mathcal{O}_1, \mathcal{O}_2)_\Sigma\}$$

$$\Phi\text{-}\,\text{AT}(\mathcal{O}_1, \mathcal{O}_2)_\Sigma = \textstyle\bigcup_{Y \in \{L, R, \top, \bot\}} \Phi\text{-}\,\text{AT}(\mathcal{O}_1, \mathcal{O}_2)^Y_\Sigma$$

Given a concept name $A \in \Phi$-$\text{AT}(\mathcal{O}_1, \mathcal{O}_2)^\text{L}_\Sigma$ (analogously $A \in \Phi$-$\text{AT}(\mathcal{O}_1, \mathcal{O}_2)^\text{R}_\Sigma$), and a set of terms $\Sigma^+ := \Sigma \cup \{\top, \bot\}$, we define the following notions:

A direct change *of A is a witness C s.t. $A \sqsubseteq C$ ($C \sqsubseteq A$) $\in \Phi\,\text{Diff}(\mathcal{O}_1, \mathcal{O}_2)$ and there is no $B \in \Sigma^+$ s.t. $\mathcal{O}_2 \models A \sqsubseteq B$ ($\mathcal{O}_2 \models B \sqsubseteq A$), $\mathcal{O}_2 \not\models A \equiv B$, and $B \sqsubseteq C$ ($C \sqsubseteq B$) $\in \Phi\,\text{Diff}(\mathcal{O}_1, \mathcal{O}_2)$.*

An indirect change *of A is a witness C s.t. $A \sqsubseteq C$ ($C \sqsubseteq A$) $\in \Phi\,\text{Diff}(\mathcal{O}_1, \mathcal{O}_2)$ and there is at least one $B \in \Sigma^+$ s.t. $\mathcal{O}_2 \models A \sqsubseteq B$ ($\mathcal{O}_2 \models B \sqsubseteq A$), $\mathcal{O}_2 \not\models A \equiv B$ and $B \sqsubseteq C$ ($C \sqsubseteq B$) $\in \Phi\,\text{Diff}(\mathcal{O}_1, \mathcal{O}_2)$.*

Concept A is purely directly changed *if it is only directly changed (analogously for* purely indirectly changed*).*

As a consequence of Definition 4, Φ-AT$(\mathcal{O}_1, \mathcal{O}_2) \subseteq$ MinCS$(\mathcal{O}_1, \mathcal{O}_2)$. Once again, take as an example the ontologies in Table 1; we have that B is purely directly specialised via witness D: $\mathcal{O}_1 \not\models B \sqsubseteq D$ and $\mathcal{O}_2 \models B \sqsubseteq D$, while A is indirectly specialised via the same witness, since $\mathcal{O}_1 \not\models A \sqsubseteq D$, $\mathcal{O}_2 \models A \sqsubseteq D$, $\mathcal{O}_2 \models A \sqsubseteq B$ and $B \sqsubseteq D \in$ Diff$(\mathcal{O}_1, \mathcal{O}_2)$. In other words, concept A changes via B. Additionally, the concept D is directly generalised via B, but indirectly generalised via A. Thus D is not purely directly changed, but rather we have a mixed effect on the concept.

The distinction between directly and indirectly affected concept names, in addition to the separation of concepts affected via $\top$ and $\bot$, allows us to overcome the problem described in Section 3, w.r.t. propositionally closed description logics. If there exists a global change to $\top$ (analogously to $\bot$), it is singled out from the remaining localised changes, and its effect is appropriately marked as an indirect change to every concept name. Thus the diff results are no longer "polluted" by vacuous witnesses such as those exemplified and discussed in Section 3. The notion of "change effect" as per Definition 4 is applicable to any diff function Φ that produces a set of witness axioms $\Phi\,$Diff$(\mathcal{O}_1, \mathcal{O}_2)_\Sigma$.

4.3 Diff Functions

Deciding the minimal change set between two ontologies involves deciding whether, for a given signature Σ, two ontologies are mCE-inseparable w.r.t. Σ. Since mCE-inseparability is undecidable for $\mathcal{SROIQ}$ [9],[2] we devise several sound but incomplete approximations to the problem of computing the minimal change set: to start with, "Subconcept" diff, denoted SubDiff$(\mathcal{O}_1, \mathcal{O}_2)_\Sigma$, and "Grammar" diff, denoted GrDiff$(\mathcal{O}_1, \mathcal{O}_2)_\Sigma$. The set of differences that would be captured by a simple comparison of concept hierarchies between two ontologies, i.e., differences in atomic subsumptions, is denoted AtDiff$(\mathcal{O}_1, \mathcal{O}_2)_\Sigma$. Hereafter we refer to the semantic diff notion used within ContentCVS as CvsDiff$(\mathcal{O}_1, \mathcal{O}_2)_\Sigma$.

The SubDiff$(\mathcal{O}_1, \mathcal{O}_2)_\Sigma$ approximation is based on subconcepts explicitly asserted in the input ontologies, and returns those differences in entailments of type $C \sqsubseteq D$, where C and D are possibly complex concepts from the set of Σ-subconcepts of $\mathcal{O}_1$ and $\mathcal{O}_2$ (see Definition 5). It is at least conceivable that many entailments will involve subconcepts, and, if that is the case, those would be witnesses that the user could understand and, indeed, may have desired. Furthermore, this notion may find entailment differences that would not show up if we restrict ourselves to either atomic subsumptions, or specific forms of entailments (in the manner of CvsDiff$(\mathcal{O}_1, \mathcal{O}_2)_\Sigma$). The restriction to forms of concepts explicit in either ontology, however, limits the amount of change captured. In our ontologies in Table 1, e.g., the change to concept D: $\mathcal{O}_1 \not\models D \sqsubseteq \exists r.\forall s.F$, while $\mathcal{O}_2 \models D \sqsubseteq \exists r.\forall s.F$, cannot be captured by SubDiff$(\mathcal{O}_1, \mathcal{O}_2)_\Sigma$. However, the rationale behind this approach is that we could detect other kinds of change in a principled and relatively cheap way, e.g., we have that $\mathcal{O}_1 \not\models A \sqsubseteq \exists r.E$, and $\mathcal{O}_2 \models A \sqsubseteq \exists r.E$. Obviously we could arbitrarily extend our entailment grammar to, for instance, a subset of the $\mathcal{SROIQ}$ closure, thus finding even

[2] Indeed mCE-inseparability is already undecidable for general $\mathcal{EL}$ ontologies [10].

more witnesses. Though our aim is to capture as much change as possible while maintaining both computational feasibility and legibility of witness axioms.

Nevertheless, in order to avoid only considering witnesses in their explicitly asserted form, we extend the previous diff notion to $\text{GrDiff}(\mathcal{O}_1, \mathcal{O}_2)_\Sigma$, which detects differences in additional types of entailments using the following grammars (where SC, SC' stand for subconcepts of $\mathcal{O}_1 \cup \mathcal{O}_2$, and r a role name):

Grammar $G_L : C \longrightarrow SC \mid SC \sqcup SC' \mid \exists r.SC \mid \forall r.SC \mid \neg SC$
Grammar $G_R : C \longrightarrow SC \mid SC \sqcap SC' \mid \exists r.SC \mid \forall r.SC \mid \neg SC$

$\text{GrDiff}(\mathcal{O}_1, \mathcal{O}_2)_\Sigma$ combines the basic intuitions about interesting logical forms with the ontology specific information available from $\text{SubDiff}(\mathcal{O}_1, \mathcal{O}_2)_\Sigma$ to be somewhat less ad hoc. By restricting fillers of the restrictions to the (inherently) finite set of subconcepts, we ensure termination. The grammars are slightly optimized to avoid pointless redundancies, such as testing for $A \sqsubseteq C \sqcap D$ which is equivalent to $A \sqsubseteq C$ and $A \sqsubseteq D$. It is not obvious how to reasonably extend these grammars to incorporate features such as number restrictions.

In terms of computational complexity, there are two dimensions to be considered: 1) the complexity of deciding entailment in the input language, and 2) the number of entailment tests. Regarding the latter, the maximum number of *candidate witness axioms* is polynomial in the number of the inputs' subconcepts, namely quadratic for $\text{SubDiff}(\mathcal{O}_1, \mathcal{O}_2)_\Sigma$ and cubic for $\text{GrDiff}(\mathcal{O}_1, \mathcal{O}_2)_\Sigma$.

The semantic difference between ontologies w.r.t. each mentioned diff function, including CEX and $\text{CvsDiff}(\mathcal{O}_1, \mathcal{O}_2)_\Sigma$, is boiled down to finding an entailment that holds in $\mathcal{O}_2$ but not $\mathcal{O}_1$; what varies between each function is the kind of entailment grammar used, which in turn dictates the computational feasibility of the diff function.

Definition 5. *Given two ontologies, a diff function Φ, and a signature Σ, the* set of Σ-differences *is:*

$$\Phi\,\text{Diff}(\mathcal{O}_1, \mathcal{O}_2)_\Sigma := \ \{\eta \in \Phi\text{-}ax \mid \mathcal{O}_1 \not\models \eta \wedge \mathcal{O}_2 \models \eta \wedge \widetilde{\eta} \subseteq \Sigma\}$$

where the set Φ-ax is defined as follows:

if $\Phi = At$, $\{C \sqsubseteq D \mid C, D \in \Sigma\}$
if $\Phi = Sub$, $\{C \sqsubseteq D \mid C, D \textit{ subconcepts in } \mathcal{O}_1 \cup \mathcal{O}_2\}$
if $\Phi = Gr$, $\{C \sqsubseteq D \mid D \textit{ a concept over } G_L, \textit{or } C \textit{ a concept over } G_R\}$
if $\Phi = Cvs$, $\{C \sqsubseteq D \mid C \in \Sigma \textit{ and } D \textit{ a concept over } G_{cvs}\}$
if $\Phi = CEX$, $\{C \sqsubseteq D \mid C, D \textit{ subconcepts in } \mathcal{L}(\Sigma)\}$

Applying the diff functions At, Sub, and Gr from Definition 5 to our example ontologies from Table 1, we get the sets of affected terms described in Table 2.

The differences in atomic subsumptions are easily identifiable, and follow from axioms α_1, α_2 in $\mathcal{O}_1$ and their β_1, β_2 counterparts in $\mathcal{O}_2$. In addition to these, $\text{SubDiff}(\mathcal{O}_1, \mathcal{O}_2)_\Sigma$ pinpoints the axioms β_4 and β_5 as new entailments in $\mathcal{O}_2$, thus concept E is regarded as specialised via β_4, and I generalised via β_5. Finally

Table 2. Affected concepts (specialised, generalised and total) between $\mathcal{O}_1$ and $\mathcal{O}_2$ according to the mentioned diff notions

	$\Phi = At$	$\Phi = Sub$	$\Phi = Gr$	$\text{MinCS}(\mathcal{O}_1, \mathcal{O}_2)^C$
$\Phi\text{-AT}(\mathcal{O}_1, \mathcal{O}_2)^L_\Sigma$	$\{A, B\}$	$\{A, B, E\}$	$\{A, B, D, E\}$	-
$\Phi\text{-AT}(\mathcal{O}_1, \mathcal{O}_2)^R_\Sigma$	$\{D\}$	$\{D, I\}$	$\{D, I, J\}$	-
$\Phi\text{-AT}(\mathcal{O}_1, \mathcal{O}_2)_\Sigma$	$\{A, B, D\}$	$\{A, B, D, E, I\}$	$\{A, B, D, E, I, J\}$	$\Sigma \setminus \{C, G\}$

$\text{GrDiff}(\mathcal{O}_1, \mathcal{O}_2)_\Sigma$ spots two more affected concepts: D is specialised via witness axiom $D \sqsubseteq \exists r.\forall s.(F \sqcap G)$, and J is generalised via $\exists r.\forall t.H \sqsubseteq J$. Taking into account $\text{MinCS}(\mathcal{O}_1, \mathcal{O}_2)^C$, it is evident that the more we expand our entailment grammar, the closer we get to the actual change set, while remaining decidable – as long as the language generated by the grammar is finite. We already discussed the computational upper bound above, and will comment on the performance of our implementation in Section 5.

It is not hard to see that there are subset relations between each diff, and the set $\text{MinCS}(\mathcal{O}_1, \mathcal{O}_2)$ that they approximate, as per Lemma 1:

Lemma 1. *Given two ontologies and a signature Σ:*

$$\text{At-AT}(\mathcal{O}_1, \mathcal{O}_2)_\Sigma \subseteq \text{Sub-AT}(\mathcal{O}_1, \mathcal{O}_2)_\Sigma \subseteq \text{Gr-AT}(\mathcal{O}_1, \mathcal{O}_2)_\Sigma \subseteq \text{MinCS}(\mathcal{O}_1, \mathcal{O}_2)$$
$$\text{Cvs-AT}(\mathcal{O}_1, \mathcal{O}_2)_\Sigma \subseteq \text{Gr-AT}(\mathcal{O}_1, \mathcal{O}_2)_\Sigma \subseteq \text{MinCS}(\mathcal{O}_1, \mathcal{O}_2)$$

As for CEX, its current implementation only takes as input acyclic $\mathcal{ELH}^r$ *terminologies*, that is, $\mathcal{ELH}^r$ TBoxes which are 1) acyclic and 2) every concept appears (alone) on the left-hand side of an axiom exactly once. In order to apply CEX to knowledge bases that are more expressive than $\mathcal{ELH}^r$ terminologies, one must rely on approximation algorithms. An $\mathcal{EL}$ approximation does not suffice, as there may exist cycles, GCIs, or more than one axiom with the same left hand side. Therefore, as a means to apply CEX to expressive ontologies, we use two $\mathcal{ELH}^r$ approximations.

Definition 6. *For an ontology $\mathcal{O}$, we define the approximation function $\mathcal{ELH}^r\text{App}_1(\mathcal{O})$ that approximates $\mathcal{O}$ into $\mathcal{ELH}^r$ as follows:*

(a) Remove all axioms with a non-atomic left hand side and all non-$\mathcal{EL}$ axioms.
(b) If there is an equivalence axiom with an atomic left or right hand side X, and a non-empty set of subsumptions Ψ that have X on their left hand side, remove all axioms in Ψ.
(c) Break cycles by non-deterministically removing axioms in cycles until the resulting ontology is cycle-free.
(d) Remove all but one axiom with a given atomic left-hand side.

The approximation function $\mathcal{ELH}^r\text{App}_2(\mathcal{O})$ is the same as $\mathcal{ELH}^r\text{App}_1(\mathcal{O})$ but with Step (d) *replaced with* (d') *as follows:*

(d') Replace the set of axioms with a common left hand side concept A, e.g., $\{A \sqsubseteq C, A \sqsubseteq D\}$, with a subsumption between A and the conjunction of all concepts on the right hand side of all such axioms, e.g., $A \sqsubseteq C \sqcap D$.

Based on these approximation algorithms, we can now use CEX as a sub-routine in a diff function for non-$\mathcal{ELH}^r$ ontologies.

Definition 7. *Given two ontologies, a signature Σ, and an $\mathcal{ELH}^r$ approximation function $\mathcal{ELH}^{\mathrm{r}}\mathrm{App}_i(\mathcal{O})$, the set of Σ-differences $\mathrm{Cex}_i\mathrm{Diff}(\mathcal{O}_1, \mathcal{O}_2)_\Sigma$ is:*

1. *For each $j \in \{1, 2\}$, execute $\mathcal{ELH}^{\mathrm{r}}\mathrm{App}_i(\mathcal{O}_j)$, resulting in $\mathcal{O}'_j$.*
2. *Apply CEX to ($\mathcal{O}'_1$, $\mathcal{O}'_2$, Σ), resulting in the change set:* TempCS.
3. *For each $\alpha \in TempCS$, add α to $\mathrm{Cex}_i\mathrm{Diff}(\mathcal{O}_1, \mathcal{O}_2)_\Sigma$ if $\mathcal{O}_1 \not\models \alpha$ and $\mathcal{O}_2 \models \alpha$.*

Given the loss of axioms during the input approximation step (via the $\mathcal{ELH}^r$ approximation functions), especially due to its non-deterministic nature, we may well introduce spurious changes. Thus Step 3 in Definition 7 is designed to ensure that changes detected within the $\mathcal{ELH}^r$ approximations (obtained in Step 2) are sound changes w.r.t. the whole (untouched) input ontologies. In other words, to verify which detected changes are due to the approximation step. Obviously, this approximation-based procedure throws away a lot of information and is not deterministic. However, even such an approximation can offer useful insight, particularly if it finds changes that other methods do not. There are more elaborate existing approximation approaches (e.g., [12]), but they generally do not produce $\mathcal{ELH}^r$ terminology, so their use requires either changing the approximation output or updating CEX to take non-terminological $\mathcal{EL}$ input.

5 Empirical Results

The object of our evaluation is a subset of the NCIt corpus used in [3], with expressivity ranging from $\mathcal{ALCH}(\mathcal{D})$ to $\mathcal{SH}(\mathcal{D})$. More specifically, we take into account 14 versions (out of 103 in the whole corpus) of the NCIt (from release 05.06f to 06.08d), and perform consecutive, pairwise comparisons between those versions which contain concept-based change logs. These versions range from ≈70,000 to ≈85,000 logical axioms, and from ≈43,000 to ≈57,000 concept names. In order to investigate the applicability of our approach we *(1)* compare the results obtained via our approximations with those output by $\mathrm{Cex}_1\mathrm{Diff}(\mathcal{O}_1, \mathcal{O}_2)_\Sigma$, $\mathrm{Cex}_2\mathrm{Diff}(\mathcal{O}_1, \mathcal{O}_2)_\Sigma$ and $\mathrm{CvsDiff}(\mathcal{O}_1, \mathcal{O}_2)_\Sigma$, *(2)* compare the number of (purely) directly and indirectly affected concepts, and, finally, *(3)* inspect whether the devised approximations capture changes not reported in the NCIt change logs.

The experiment machine used is an Intel Xeon Quad-Core 3.20GHz with 16Gb DDR3 RAM. The system runs Mac OS X 10.6.8, Java Virtual Machine (JVM v1.5), and all tests were run using the OWL API (v3.2.4) [4].

Table 3. Number of concepts processed per minute by each diff function Φ

	$\Phi = \text{Cex}_1$	$\Phi = \text{Cex}_2$	$\Phi = At$	$\Phi = Sub$	$\Phi = Cvs$	$\Phi = Gr$
#Concepts/Min.	151	143	13,547	127	58	50

In terms of computation times, the average number of concepts processed per minute by each diff function is shown in Table 3.[3] The typical total time ranges from seconds for $\text{AtDiff}(\mathcal{O}_1, \mathcal{O}_2)_\Sigma$, to about 30 minutes for the $\text{Cex}_1\text{Diff}(\mathcal{O}_1, \mathcal{O}_2)_\Sigma$, $\text{Cex}_2\text{Diff}(\mathcal{O}_1, \mathcal{O}_2)_\Sigma$, and $\text{Sub-AT}(\mathcal{O}_1, \mathcal{O}_2)_\Sigma$ to hours for $\text{GrDiff}(\mathcal{O}_1, \mathcal{O}_2)_\Sigma$ and $\text{CvsDiff}(\mathcal{O}_1, \mathcal{O}_2)_\Sigma$.

$\text{GrDiff}(\mathcal{O}_1, \mathcal{O}_2)_\Sigma$ and $\text{CvsDiff}(\mathcal{O}_1, \mathcal{O}_2)_\Sigma$ are rather computationally expensive; the current implementation uses a naive "generate-and-test" approach, where, for each concept, we generate candidate witnesses from the grammar until a witness is found or we exhaust the set. There is clearly considerable scope for optimization.

5.1 Diff Comparison

The comparison of each diff w.r.t. the total number of affected concept names found is presented in Table 4. Figure 1 shows a comparison of the number of affected concept names found by $\text{Cvs-AT}(\mathcal{O}_1, \mathcal{O}_2)_\Sigma$ and $\text{Gr-AT}(\mathcal{O}_1, \mathcal{O}_2)_\Sigma$ within the randomly selected signatures. Due to computational issues regarding $\text{GrDiff}(\mathcal{O}_1, \mathcal{O}_2)_\Sigma$ and $\text{CvsDiff}(\mathcal{O}_1, \mathcal{O}_2)_\Sigma$, instead of comparing each pair of NCIt versions w.r.t. Σ_u, we take a random sample of the terms in Σ_u (generally n $\approx$ 1800) such that a straightforward extrapolation allows us to determine that the true proportion of changed terms in Σ_u lies in the confidence interval (+-3%) with a 99% confidence level.

In general, $\text{Gr-AT}(\mathcal{O}_1, \mathcal{O}_2)_\Sigma$, even taking into account the confidence interval, consistently detects more affected concepts (both L and R, i.e., specialised and generalised, accordingly) than all other diffs. The CEX-based approximation $\text{Cex}_1\text{-AT}(\mathcal{O}_1, \mathcal{O}_2)_\Sigma$ performs poorly across the board, consistently capturing less affected concepts than even a comparison of atomic subsumptions. The second CEX-based approximation $\text{Cex}_2\text{-AT}(\mathcal{O}_1, \mathcal{O}_2)_\Sigma$, however, typically detects more affected terms than $\text{At-AT}(\mathcal{O}_1, \mathcal{O}_2)_\Sigma$, apart from two cases ($d_6$ and d_{10}), but still less than $\text{Sub-AT}(\mathcal{O}_1, \mathcal{O}_2)_\Sigma$. Regardless of this result, it is not the case that $\text{Sub-AT}(\mathcal{O}_1, \mathcal{O}_2)_\Sigma$ is always better than $\text{Cex}_2\text{-AT}(\mathcal{O}_1, \mathcal{O}_2)_\Sigma$, as the latter actually detects more generalised concept names than $\text{Sub-AT}(\mathcal{O}_1, \mathcal{O}_2)_\Sigma$ in all but one case. The gathered evidence suggests that indeed combining these approaches would perhaps result in a preferable semantic diff solution than one or the other, as exhibited by the higher average coverage of 59% in $\text{Un-AT}(\mathcal{O}_1, \mathcal{O}_2)_\Sigma$ (although only an 8% increase w.r.t. to the average coverage of $\text{Sub-AT}(\mathcal{O}_1, \mathcal{O}_2)_\Sigma$). As expected, $\text{Gr-AT}(\mathcal{O}_1, \mathcal{O}_2)_\Sigma$ captures more specialised

[3] Note that, originally, $\text{CvsDiff}(\mathcal{O}_1, \mathcal{O}_2)_\Sigma$ only computes $\text{AT}(\mathcal{O}_1, \mathcal{O}_2)^{\text{L}}_{\Sigma}$, but in order to provide a direct comparison with the diffs here proposed we also compute $\text{AT}(\mathcal{O}_1, \mathcal{O}_2)^{\text{R}}_{\Sigma}$ according to the G_{cvs} grammar.

Table 4. Number of affected concept names, $\text{AT}(\mathcal{O}_i, \mathcal{O}_{i+1})_\Sigma$, found by each diff function (in addition to Un-AT := $\{\text{Cex}_1\text{-AT} \cup \text{Cex}_2\text{-AT} \cup \text{Sub-AT}\}$) w.r.t. $\Sigma := \Sigma_u$, and their respective coverage w.r.t. $\text{Gr-AT}(\mathcal{O}_i, \mathcal{O}_{i+1})_\Sigma$. At this point, no distinction is made between direct and indirect changes.

Comparison	Cex_1-AT	Cex_2-AT	At-AT	Sub-AT	Un-AT	Gr-AT
d_1	1,134	1,922	1,416	2,131	3,311	43,096
d_2	877	1,746	1,208	1,816	3,307	43,928
d_3	5,415	6,287	6,135	6,528	8,818	45,639
d_4	2,145	6,198	3,676	45,932	45,932	46,929
d_5	3,964	7,656	4,978	15,691	15,758	48,075
d_6	2,298	3,718	3,923	6,203	8,570	48,629
d_7	1,893	3,393	3,217	6,330	7,508	49,189
d_8	6,387	7,397	6,806	7,428	8,957	54,870
d_9	1,655	4,460	2,745	5,329	6,913	55,555
d_{10}	1,512	3,681	4,553	6,415	8,147	55,948
d_{11}	1,102	3,026	1,714	4,325	5,916	57,036
Avg. Cov.	18%	23%	27%	55%	59%	
Min. Cov.	3%	8%	5%	18%	21%	
Max. Cov.	47%	49%	52%	100%	100%	

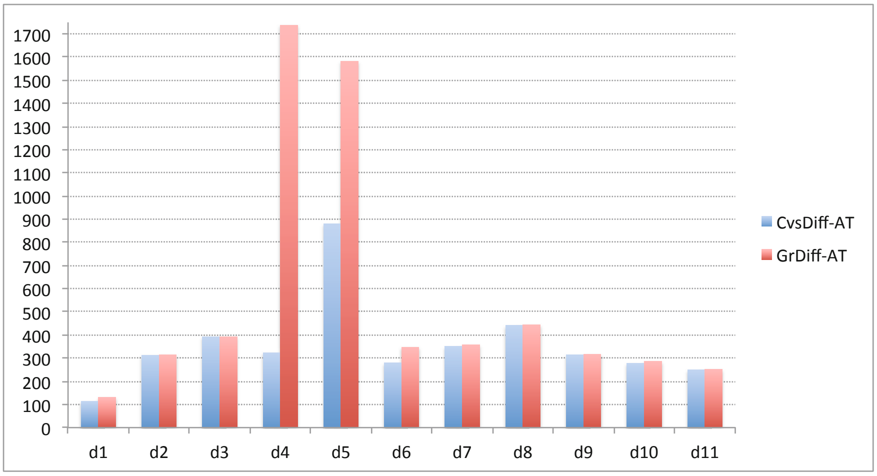


Fig. 1. Comparison of number of specialised concepts found by $\text{Cvs-AT}(\mathcal{O}_1, \mathcal{O}_2)_\Sigma$ and $\text{Gr-AT}(\mathcal{O}_1, \mathcal{O}_2)_\Sigma$ within the signature samples of the NCIt (y-axis: number of concept names, x-axis: comparison identifier)

concepts than $\text{Cvs-AT}(\mathcal{O}_1, \mathcal{O}_2)_\Sigma$ in all cases, evidenced in Figure 1.[4] Both of these diff approaches resolve all terms in the random signature as generalised.

[4] Note that, since $\text{Gr-AT}(\mathcal{O}_1, \mathcal{O}_2)^R_\Sigma = \text{Cvs-AT}(\mathcal{O}_1, \mathcal{O}_2)^R_\Sigma$, we only present in Figure 1 the results of $\text{Gr-AT}(\mathcal{O}_1, \mathcal{O}_2)^L_\Sigma$ and $\text{Cvs-AT}(\mathcal{O}_1, \mathcal{O}_2)^L_\Sigma$.

Thus the projected value implies that nearly, if not every term in the full signature has been generalised.

5.2 Splitting Direct and Indirect Changes

Having the results of each diff at hand, i.e., the set of affected concepts and, for each of these, the set of witnesses, we can then tell apart those concept names that are directly, indirectly, or both directly and indirectly affected. Note that, as an optimisation within the implementation of $\text{GrDiff}(\mathcal{O}_1, \mathcal{O}_2)_\Sigma$ and $\text{CvsDiff}(\mathcal{O}_1, \mathcal{O}_2)_\Sigma$ diff, we only compute one witness per concept. Thus we do not possess the full set of witnesses, making the distinction of directly and indirectly affected concepts possibly unsound and incomplete. As such, we apply this distinction only to $\text{SubDiff}(\mathcal{O}_1, \mathcal{O}_2)_\Sigma$ and $\text{AtDiff}(\mathcal{O}_1, \mathcal{O}_2)_\Sigma$. Figure 2 shows the total number of purely direct, purely indirect, and both directly and indirectly affected concepts found within $\text{At-AT}(\mathcal{O}_1, \mathcal{O}_2)_\Sigma$ and $\text{Sub-AT}(\mathcal{O}_1, \mathcal{O}_2)_\Sigma$. Note that the size of $\text{SubDiff}(\mathcal{O}_1, \mathcal{O}_2)_\Sigma$ can be smaller than $\text{AtDiff}(\mathcal{O}_1, \mathcal{O}_2)_\Sigma$, as in versions v3 and v4. For these particular cases, we bring to the front the smaller value (i.e. $\text{SubDiff}(\mathcal{O}_1, \mathcal{O}_2)_\Sigma$), and the value of $\text{AtDiff}(\mathcal{O}_1, \mathcal{O}_2)_\Sigma$ becomes

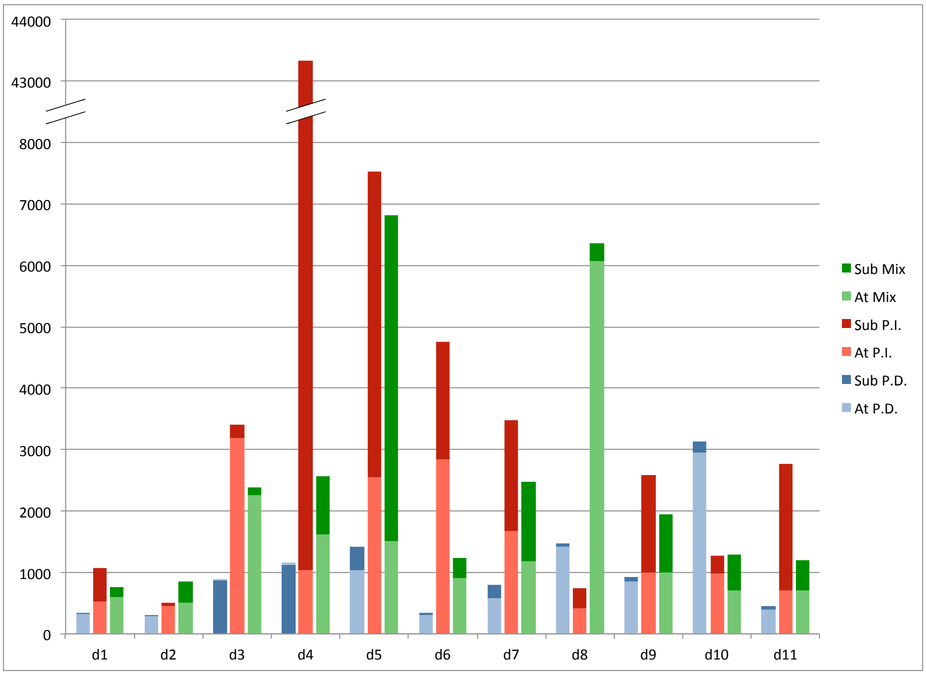


Fig. 2. Comparison of purely directly ("P.D."), purely indirectly ("P.I."), and both directly and indirectly (denoted "Mix") affected concepts found within $\text{At-AT}(\mathcal{O}_1, \mathcal{O}_2)_\Sigma$ (denoted "At"), and $\text{Sub-AT}(\mathcal{O}_1, \mathcal{O}_2)_\Sigma$ (denoted "Sub") in NCIt versions (y-axis: number of concept names, x-axis: comparison identifier)

the increment. Also, this figure presents the total number of changes; the union of $\text{AT}(\mathcal{O}_1, \mathcal{O}_2)^{\text{L}}_{\Sigma}$ and $\text{AT}(\mathcal{O}_1, \mathcal{O}_2)^{\text{R}}_{\Sigma}$.

In general, the number of purely directly changed concepts is much smaller than the number of purely indirect or mixed. One case is particularly surprising: $\text{Sub-AT}(\mathcal{O}_1, \mathcal{O}_2)_{\Sigma}$ contains 43,326 purely indirect changes in v4, and only 1,122 purely direct ones. In an ontology engineering scenario, where one or more people are required to analyse such change sets, having this mechanism for isolating changes of most interest is conceivably a preferable means to analyse a change set, in addition to providing a basis for producing more intelligible change logs with impact analysis.

5.3 Analysis of the NCIt Change Logs

The change logs supplied with each version of the NCIt contain those concept names which were subject to changes. However, it is unclear whether each reported change may also (or solely) relate to annotation changes. It could be the case that a reported concept change is purely ineffectual as well. In spite of this ambiguity, it should be expected that a change log contains at least those concept names that are directly changed, and this is what we aim to find out in our next experiment; we extract the concept names mentioned in the change log, and verify whether the obtained direct changes for each NCIt comparison are contained in said change logs. The results are shown in Table 5, comparing the number of directly affected concept names found within $\text{At-AT}(\mathcal{O}_1, \mathcal{O}_2)_{\Sigma}$ and $\text{Sub-AT}(\mathcal{O}_1, \mathcal{O}_2)_{\Sigma}$, and how many of those are not present n the NCIt change logs. Overall, we determined that the change logs are missing a lot of direct

Table 5. Number of directly affected concepts *1)* in $\text{AT}(\mathcal{O}_1, \mathcal{O}_2)^{\text{L}}_{\Sigma}$ (denoted "L"), *2)* in $\text{AT}(\mathcal{O}_1, \mathcal{O}_2)^{\text{R}}_{\Sigma}$ (denoted "R"), *3)* in the union of those two sets (denoted "Total"), and *4)* that do not appear in the NCIt change logs (denoted "Missed in Log"), found by $\text{AtDiff}(\mathcal{O}_1, \mathcal{O}_2)_{\Sigma}$ and $\text{SubDiff}(\mathcal{O}_1, \mathcal{O}_2)_{\Sigma}$ for $\Sigma := \Sigma_u$

NCIt	$\text{At-AT}(\mathcal{O}_1, \mathcal{O}_2)_{\Sigma}$				$\text{Sub-AT}(\mathcal{O}_1, \mathcal{O}_2)_{\Sigma}$			
version	L	R	Total	Missed in Log	L	R	Total	Missed in Log
d_1	646	294	896	798	820	298	1,060	953
d_2	565	274	772	149	1,147	294	1,298	211
d_3	2,321	891	2,991	315	2,791	898	3,090	445
d_4	1,624	1,187	2,683	190	2,725	1,198	2,814	432
d_5	1,555	1,009	2,465	243	8,038	1,186	9,142	317
d_6	890	385	1,130	199	1,306	401	1,485	199
d_7	1,190	704	1,637	273	2,720	780	2,935	511
d_8	6,075	1,421	6,389	5,546	6,411	1,465	6,693	5,723
d_9	1,481	420	1,766	207	2,607	478	2,782	322
d_{10}	3,321	370	3,579	216	4,964	427	5,217	298
d_{11}	753	378	1,043	300	1,404	472	1,643	582
Total	20421	7,333	25,351	8,436	34,933	7,897	38,159	9,993

changes. More specifically, on average, At-AT$(\mathcal{O}_1, \mathcal{O}_2)_\Sigma$ contains 767 directly affected concept names not mentioned in the change logs, while Sub-AT$(\mathcal{O}_1, \mathcal{O}_2)_\Sigma$ uncovers 908 such concept names per NCIt comparison.

Subsequently we verify whether the affected concepts in Cex$_1$-AT$(\mathcal{O}_1, \mathcal{O}_2)_\Sigma$, Cex$_2$-AT$(\mathcal{O}_1, \mathcal{O}_2)_\Sigma$, At-AT$(\mathcal{O}_1, \mathcal{O}_2)_\Sigma$ and Sub-AT$(\mathcal{O}_1, \mathcal{O}_2)_\Sigma$ are contained in the NCIt change logs. This is presented in Table 6. Overall we see that none of the diffs captures the exact number of reported concept changes in the logs. The maximum coverage of the change log occurs in comparisons d_4 and d_5, where Sub-AT$(\mathcal{O}_1, \mathcal{O}_2)_\Sigma$ captures 96% and 91% of the concept names mentioned in the logs, accordingly. By taking the union of affected concepts found by the CEX-based approximations and Sub-AT$(\mathcal{O}_1, \mathcal{O}_2)_\Sigma$, the average coverage of the change logs increases to 73%.

Table 6. Number of affected concept names, AT$(\mathcal{O}_i, \mathcal{O}_{i+1})_\Sigma$, found by each diff function (in addition to Un-AT := {Cex$_1$-AT ∪ Cex$_2$-AT ∪ Sub-AT}) w.r.t. $\Sigma := \Sigma_u$ within the NCIt change logs

NCIt Version	Change Log	Cex$_1$-AT	Cex$_2$-AT	At-AT	Sub-AT	Un-AT
d_1	2,159	107	168	103	126	269
d_2	1,399	520	773	725	974	1,013
d_3	4,234	2,497	2,973	3,102	3,148	3,150
d_4	8,447	1,327	1,598	2,734	8,117	8,117
d_5	3,847	1,595	2,655	2,602	3,503	3,504
d_6	2,470	866	1,147	1,141	1,312	1,406
d_7	5,302	1,217	1,253	1,982	2,668	2,699
d_8	2,556	688	885	875	993	1,003
d_9	3,945	1,060	2,205	1,878	2,530	2,755
d_{10}	6,046	978	3,824	3,551	4,076	6,046
d_{11}	2,065	628	764	853	1,091	1,168
Avg. Coverage		27%	43%	46%	67%	73%

6 Discussion

First thing to notice is that Sub-AT$(\mathcal{O}_1, \mathcal{O}_2)_\Sigma$ finds more affected concepts than At-AT$(\mathcal{O}_1, \mathcal{O}_2)_\Sigma$, Cex$_1$-AT$(\mathcal{O}_1, \mathcal{O}_2)_\Sigma$, and Cex$_2$-AT$(\mathcal{O}_1, \mathcal{O}_2)_\Sigma$, while often not reaching close to the projected values of Gr-AT$(\mathcal{O}_1, \mathcal{O}_2)_\Sigma$ (the average coverage being 55%). The latter captures more specialised concepts within the selected signatures than Cvs-AT$(\mathcal{O}_1, \mathcal{O}_2)_\Sigma$, while the number of generalised concepts is the same for both diffs (i.e., the full signature).

Considering the high number of affected concepts in Sub-AT$(\mathcal{O}_1, \mathcal{O}_2)_\Sigma$ on comparisons d_4 and d_5 of the NCIt, one can argue that analysing such a change set would be difficult. By categorising concept names in the change set according to whether they are directly or indirectly affected, we get a succinct representation of a change set, thus significantly reducing information overload. Note that, e.g., in d_4 there are 45,825 specialised concepts, out of which there are only 78 purely directly specialised concepts, and the majority of the remainder are purely indirectly specialised concepts (43,100). Similarly in d_5, from 15,254 specialised concepts there are only 1,527 purely direct specialisations. Immediately we see that this mechanism can provide an especially helpful means to 1) assist change analysis, by, e.g., confining the changes shown upfront to only (purely) direct ones, and 2) generate more informative concept-based change logs.

7 Conclusions

We have formulated the problem of finding the set of affected terms between ontologies via model inseparability, and presented feasible approximations to finding this set. We have shown that each of the approximations can find considerably more changes than those visible in a comparison of concept hierarchies. Both sound approximations devised capture more changes than the CEX-based approximations. The restrictions imposed by CEX on the input ontologies make change-preserving approximations a challenge, as we have seen in our attempt to reduce the NCIt to $\mathcal{EL}$ in a less naive way.

The proposed distinction between (purely) direct and indirect changes allows users to focus on those changes which are specific to a given concept, in addition to masking possibly uninteresting changes to any and all concept names (such as those obtained via witnesses constructed with negation and disjunction), thereby making change analysis more straightforward. As demonstrated by the NCIt change log analysis, we have found a (often high) number of direct changes that are not contained in the NCIt change logs, which leads us to believe the recording of changes does not seem to follow from even a basic concept hierarchy comparison, but rather a seemingly ad hoc mechanism.

In future work we aim to optimise the devised approximations so as to compare all NCIt versions w.r.t. their signature union, and combine the information from this concept oriented diff with our axiom oriented one.

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. Ghilardi, S., Lutz, C., Wolter, F.: Did I damage my ontology? A case for conservative extensions in description logics. In: Proc. of KR 2006 (2006)
3. Gonçalves, R.S., Parsia, B., Sattler, U.: Categorising logical differences between OWL ontologies. In: Proc. of CIKM 2011 (2011)
4. Horridge, M., Bechhofer, S.: The OWL API: A Java API for working with OWL 2 ontologies. In: Proc. of OWLED 2009 (2009)
5. Jiménez-Ruiz, E., Cuenca Grau, B., Horrocks, I., Berlanga Llavori, R.: Supporting concurrent ontology development: Framework, algorithms and tool. Data and Knowledge Engineering 70(1), 146–164 (2011)

6. Konev, B., Walther, D., Wolter, F.: The Logical Difference Problem for Description Logic Terminologies. In: Armando, A., Baumgartner, P., Dowek, G. (eds.) IJCAR 2008. LNCS (LNAI), vol. 5195, pp. 259–274. Springer, Heidelberg (2008)
7. Křemen, P., Šmíd, M., Kouba, Z.: OWLDiff: A practical tool for comparison and merge of OWL ontologies. In: Proc. of DEXA 2012 (2011)
8. Küsters, R.: Non-Standard Inferences in Description Logics. LNCS (LNAI), vol. 2100. Springer, Heidelberg (2001)
9. Lutz, C., Walther, D., Wolter, F.: Conservative extensions in expressive description logics. In: Proc. of IJCAI 2007 (2007)
10. Lutz, C., Wolterinst, F.: Conservative Extensions in the Lightweight Description Logic $\mathcal{EL}$. In: Pfenning, F. (ed.) CADE 2007. LNCS (LNAI), vol. 4603, pp. 84–99. Springer, Heidelberg (2007)
11. Malone, J., Holloway, E., Adamusiak, T., Kapushesky, M., Zheng, J., Kolesnikov, N., Zhukova, A., Brazma, A., Parkinson, H.E.: Modeling sample variables with an experimental factor ontology. Bioinformatics 26(8), 1112–1118 (2010)
12. Ren, Y., Pan, J.Z., Zhao, Y.: Soundness Preserving Approximation for TBox Reasoning. In: Proc. of AAAI 2010 (2010)
13. Sattler, U., Schneider, T., Zakharyaschev, M.: Which kind of module should I extract? In: Proc. of DL 2009 (2009)

SPLODGE: Systematic Generation of SPARQL Benchmark Queries for Linked Open Data

Olaf Görlitz, Matthias Thimm, and Steffen Staab

Institute for Web Science and Technology
University of Koblenz-Landau, Germany
{goerlitz,thimm,staab}@uni-koblenz.de
http://west.uni-koblenz.de/

Abstract. The distributed and heterogeneous nature of Linked Open Data requires flexible and federated techniques for query evaluation. In order to evaluate current federation querying approaches a general methodology for conducting benchmarks is mandatory. In this paper, we present a classification methodology for federated SPARQL queries. This methodology can be used by developers of federated querying approaches to compose a set of test benchmarks that cover diverse characteristics of different queries and allows for comparability. We further develop a heuristic called SPLODGE for automatic generation of benchmark queries that is based on this methodology and takes into account the number of sources to be queried and several complexity parameters. We evaluate the adequacy of our methodology and the query generation strategy by applying them on the 2011 billion triple challenge data set.

1 Introduction

The Linked Data cloud offers a huge amount of machine readable, structured data and its full potential can only be leveraged by querying over multiple data sources. Therefore, efficient query processing on the Linked Data cloud is currently an active research area and different novel optimization approaches have been published [1,9,15,29,31]. As there is currently no common benchmark for federated Linked Data query evaluation, different datasets and different queries are being employed for the evaluation, which often prevents a direct comparison of approaches. A common benchmark based on actual queries on the Linked Data cloud could solve this problem. But since applications for Linked Data query processing are not in wide use yet such query collections are currently not available. Hence, the use of real queries in a Linked Data benchmark is not a viable option anytime soon.

Benchmarks serve different purposes. A major objective is to compare the performance of different implementations. Moreover, the quality of a system can be assessed by testing common cases and corner cases, e. g. queries with high complexity or queries which generate large intermediate result sets. Artificial datasets are usually highly structured [6] and allow for a well controlled evaluation environment. But an adaptation mimicking the Linked Data characteristics is not straightforward. Besides, benchmarks like $\mathsf{SP^2Bench}$ [27], LUBM [10], or BSBM [4] are typically designed for evaluating centralized RDF stores and we consider them inappropriate for the evaluation of query processing across Linked Data. Evaluation approaches based on real data can use hand-crafted queries, like in FedBench [26], or automatically generated queries as in [11]. Either way, the queries should expose characteristics which are assumed to cover a sufficiently large variety of real queries. However, meaningful queries can only be generated manually with a lot of effort because the content of the data sources needs to be analyzed in advance. In contrast, automatic query generation is less tedious and can produce many queries with specific characteristics, even for varying data sources as in the Linked Data cloud.

In this paper we abstract from specific query interfaces, i. e. query processing could be based on SPARQL endpoints, URI resolution, or the integration of data dumps. Our contribution is a methodology and a toolset for the systematic generation of SPARQL queries which cover a wide range of possible requests on the Linked Data cloud. A classification of query characteristics provides the basis for the query generation strategy. The query generation heuristic of SPLODGE (SPARQL Linked Open Data Query Generator) employs stepwise combination of query patterns which are selected based on predefined query characteristics, e. g. query structure, result size, and affected data sources. Constant values are randomly chosen and a verification step checks that all constraints are met.

In the following, we start with a review of related work. Then, in Section 3, we provide some necessary background information on both RDF and SPARQL. In Section 4 we conduct a thorough investigation of query characteristics in the context of Linked Data and, in Section 5, we continue with the presentation of our query generation approach SPLODGE. We give some insights on the implementation of our system in Section 6 and evaluate our approach in Section 7. In Section 8 we conclude with a summary and some final remarks.

2 Related Work

Federated SPARQL query processing is receiving more attention lately and a number of specific approaches have already been published, e. g. [30,23,11,25,12,15,13,29,9]. Stuckenschmidt et al. [30] employ indices for matching path patterns in queries while Harth et al. [11] use a (compressed) index of subjects, predicates, and objects in order to match queries to sources. DARQ [23] and SPLENDID [9] make use of statistical information (using hand-crafted data source descriptions or VOID [2], respectively) rather than (indices of) the content itself. FedX [29] focuses on efficient query execution techniques using chunked semi-joins. Without any precomputed statistics and a source selection based on SPARQL `ASK` queries it solely relies on join order heuristics for the query optimization. Recent work by Buil-Arada et al. [5] investigates the complexity and optimization of SPARQL 1.1 federation queries where data sources are already assigned to query expressions.

The evaluation of the above approaches is usually conducted with artificial datasets or real datasets using hand-crafted queries. LUBM [10] was one of the first benchmarks for evaluating (centralized) RDF triple store implementations. It allows for generating synthetic datasets of different sizes representing relations between entities from the university domain. The Berlin SPARQL Benchmark (BSBM) [4] and the SP^{2}Bench [27] are more recent benchmarks, as well based on scalable artificial datasets. BSBM is centered around product data and the benchmark queries mimic real user interaction. SP^{2}Bench employs the DBLP publications schema. Its data generator ensures specific characteristics of the data distribution while the benchmark queries include also less common and complex expressions like `UNION`, `FILTER`, and `OPTIONAL`. LUBM, BSBM, and SP^{2}Bench are hardly applicable for benchmarking federation systems because the data is very structured and Linked Data characteristics can not be achieved through data partitioning.

Benchmarking with real Linked Data is, for example, provided by FedBench [26]. It employs preselected datasets from the Linked Data cloud, e. g. life science and cross domain. Different query characteristics are covered with common and complex query pattern which yield in some cases many hundred thousand results. However, due to the limitation to a few hand-picked datasets and queries, FedBench lacks scalability with respect to the Linked Data cloud. DBPSB [17] employs benchmark queries which are derived from query logs of the official DBpedia endpoint. All queries are normalized, clustered, and the most frequent query patterns, including `JOIN`, `UNION`, `OPTIONAL`, solution modifiers, and filter conditions, are used as basis for a variable set of benchmark queries. It remains open if the queries, which cover only DBpedia, are representative for Linked Data. LIDAQ [31] provides benchmark queries based on crawled Linked Data. The query complexity, using either star-shaped or path-shaped join patterns, is limited to a maximum of three joins. Other query operators or additional solution modifiers are not considered. The query generator produces sets of similar queries by doing random walks of certain breadth or depth. DBPSB and LIDAQ do not consider result size or number of data sources in their query generation.

3 Background

The *Resource Description Framework* RDF is the core data representation format in the Linked Data cloud. Let U be a set of URIs, L a set of literals and B a set of blank nodes as defined in [14] with U, L and B being pairwise disjoint. The sets U, L, and B provide the vocabulary for representing knowledge according to the guidelines for publishing Linked Open Data [3]. The basic concept of knowledge representation with RDF is the *RDF triple* or *RDF statement*.

Definition 1 (RDF statement, RDF graph). *An* RDF statement *is a triple* $S \in (U \cup B) \times U \times (U \cup L \cup B)$. *An* RDF graph $\mathcal{G}$ *is a finite set of RDF statements. For an RDF statement* $S = (s, p, o)$ *the element* s *is called* subject*,* p *is called* predicate*, and* o *is called* object*.*

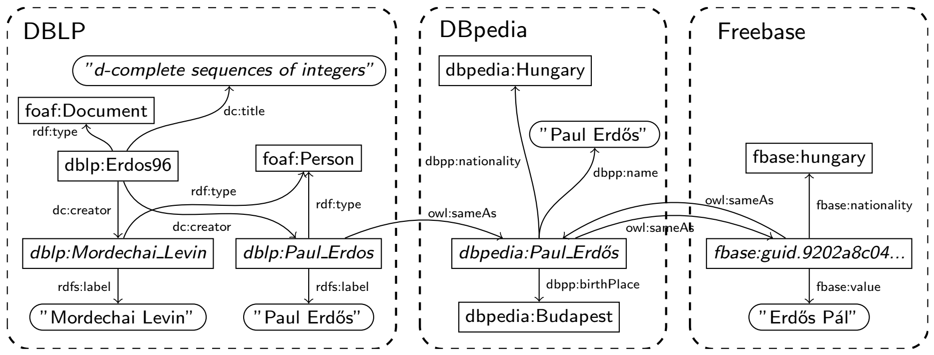
Example 1. A listing of RDF statements describing a publication by Paul Erdös (namespace definitions are omitted for better readability).

```
dblp:ErdosL96 rdf:type foaf:Document.
dblp:ErdosL96 dc:title "d−complete sequences of integers".
dblp:ErdosL96 dc:creator dblp:Paul_Erdos.
dblp:ErdosL96 dc:creator dblp:Mordechai_Levin.
dblp:Paul_Erdos rdf:type foaf:Person.
dblp:Paul_Erdos foaf:name "Paul Erdos".
dblp:Mordechai_Levin foaf:name "Mordechai Levin".
dblp:Mordechai_Levin rdf:type foaf:Person.
```

In this paper, we are interested in settings where RDF statements are distributed over a (possible large) set of different sources.

Definition 2 (Federated RDF Graph). *A* federated RDF graph $\mathcal{F}$ *is a finite set* $\mathcal{F} = \{\mathcal{G}_1, \ldots, \mathcal{G}_n\}$ *with RDF graphs* $\mathcal{G}_1, \ldots, \mathcal{G}_n$*. Let* $\mathcal{F} = \{\mathcal{G}_1, \ldots, \mathcal{G}_n\}$ *be a federated RDF graph. By abusing notation, we sometimes write* $(s, p, o) : \mathcal{G}$ *to denote that* $(s, p, o) \in \mathcal{G}$ *for* $\mathcal{G} \in \mathcal{F}$*.*

Example 2. We extend Example 1 (as illustrated in [8]) with RDF statements distributed across three Linked Data sources.



The *SPARQL Protocol and RDF Query Language* (or simply SPARQL) [22] is the standard query language for RDF graphs. The core notion of SPARQL are *graph patterns*. Let V be a set of variables disjoint from both U and L and $E(V)$ the set of *filter expressions* on V, cf. [22].

Definition 3 (Graph Patterns). *A* triple pattern *is a triple in* $(U \cup B \cup V) \times (U \cup V) \times (U \cup L \cup V)$ *and a* basic graph pattern *is a set of triple patterns. Every* basic graph pattern *is also a* graph pattern*. If* P_1, P_2 *are graph patterns and* $E \in E(V)$ *then* P_1 `UNION` P_2*,* P_1 `OPTIONAL` P_2*, and* P_1 `FILTER` E *are graph patterns as well.*

A basic graph pattern consisting of one or more triple patterns is a template that is matched in an RDF graph if all triple patterns are satisfied. Furthermore, the `UNION` combination of two patterns match if any of the two pattern matches. The `OPTIONAL` pattern matches if its first pattern matches. Additionally, further variables might get bound if the second pattern matches as well. The FILTER pattern matches if the first pattern matches and the filter expression is satisfied.

SPARQL supports four different types of queries, namely SELECT, ASK, CONSTRUCT, and DESCRIBE queries. Their main difference is the format of the query result. Given a graph pattern P and a set $\boldsymbol{x} \subseteq V$ the query SELECT $\boldsymbol{x}$ WHERE P returns tuples of variable bindings for $\boldsymbol{x}$ such that the graph pattern P, with variables substituted accordingly, is present in the queried RDF graph. A query ASK P returns true iff the graph pattern P is satisfiable (with some variable bindings). A recent study [20] of query logs of the official DBpedia endpoint revealed that the number of CONSTRUCT and DESCRIBE queries is not significant. Therefore, we will ignore those query types in this paper. Let $BGP(P)$ be the set of basic graph patterns and $TP(P)$ be the set of triple patterns appearing in a graph pattern P. For a graph pattern P and an RDF graph $\mathcal{R}$ let $eval(P, \mathcal{G})$ be the set of possible assignments of the variables in P — i. e. functions σ of the form $\sigma : V \to U \cup B \cup L$ — such that the resulting graph pattern is satisfied in $\mathcal{G}$. We refer the interested reader to [22] for the complete semantics of SPARQL.

The SPARQL federation extension [21] introduces the two keywords SERVICE and BINDINGS and enables SPARQL queries on federated RDF graphs. While the SERVICE keyword is used for specifying RDF graphs to be queried within the federated system, the BINDINGS keyword provides means for passing the values of bounded variables to sub-queries on other RDF graphs. However, this extension only allows for the specification of federated queries when the individual RDF graphs are known to be able to answer the given sub-queries. The task of determining which RDF graphs to ask for certain sub-queries is outside the scope of the federation extension but has to be addressed by other mechanisms.

4 Parameterizing Queries

At the end of the previous section we pointed out that SPARQL querying in federated environments poses serious demands on distributed querying techniques. In order to evaluate approaches for federated SPARQL processing, multi-source queries are of major interest. Moreover, SPARQL queries used for evaluation are typically classified and parametrized along several further dimensions, e. g. with respect to complexity. However, due to the sparsity of the Linked Data cloud [24] there are today only a few real-world queries that show these characteristics. For an objective evaluation and comparison with state-of-the-art systems the developer of a federated query processing system, however, needs real data and realistic SPARQL queries. Therefore, specifically designed evaluation queries should cover common characteristics of real queries and include a sufficiently large set of SPARQL features. In the following, we develop a methodology and a toolkit that can be used by developers of approaches to federated query processing for evaluating their system in a reproducible and comparable manner. There, a developer's first step is to select and combine query parameters to fit the desired evaluation scenario, e. g. common queries and corner cases. The next step is the query generation according to the defined parameters. Finally, the evaluation is conducted and results are presented. In order to make an evaluation reproducible, the chosen parameters and queries should be disclosed as well. In the following, we discuss different properties of SPARQL queries including aspects of distributed query processing.

We studied analysis results of real SPARQL queries [20,16,7] and features of RDF benchmarks [10,4,27,11,17] to compile query characteristics (i. e. properties) which we consider important for a query processing benchmark on Linked Data. One may argue about the choice of query characteristics. However, we do not claim completeness and like to encourage extensions of the classification parameters.

The first set of query characteristics relate to the semantic properties of SPARQL, i. e. the *Query Algebra.*

Query Type. SPARQL supports four query types, namely `SELECT`, `CONSTRUCT`, `ASK`, and `DESCRIBE`. They define query patterns in a `WHERE` clause and return a multiset of variable bindings, an RDF graph, or a boolean value, respectively. `DESCRIBE` queries can also take a single URI and return an RDF graph.

Join Type. SPARQL supports different join types, i. e. *conjunctive join* (`.`), *disjunctive join* (`UNION`), and *left-join* (`OPTIONAL`). These joins imply a different complexity concerning the query evaluation [28].

Result Modifiers. `DISTINCT`, `LIMIT`, `OFFSET`, and `ORDER_BY` alter the result set which is returned. They also increase the complexity for the query evaluation.

The next properties deal with the *Query Structure*, i. e. how basic graph patterns are combined in a complex graph pattern.

Variable Patterns. There are eight different combinations for having zero to three variables in subject, predicate, or object position of an RDF triple pattern. Some of these combinations, like bound predicate with variables in subject and/or objection position, are more common than others.

Join Patterns. Joins are defined by using the same variable in different triple patterns of a basic graph pattern. Typical join combinations are subject-subject joins (star shape) and subject-object joins (path shape). The combination of star-shaped and path-shaped joins yields a hybrid join pattern.

Cross Products. Conjunctive joins over triple patterns which do not share a common variable imply cross products. While the join parts can be evaluated independently, the cross product may involve large intermediate result sets.

The third group of properties deals with *Query Cardinality*, i. e. the number of sources, the number or joins, and the result size. Following, let P be a graph pattern, $\mathcal{F}$ be a federated RDF graph, and let $\mathcal{F}_{\mathcal{P}} \subseteq \mathcal{F}$ the set of *relevant data sources* for P, i. e. $\mathcal{F}_P = \{\mathcal{G} \in \mathcal{F} \mid \exists\ S \in TP(Q) : eval(S, \mathcal{G}) \neq \emptyset\,\}$.

Number of Sources. Our benchmark methodology is designed for query execution across Linked Data. Therefore, the number of data sources involved in query answering is an important factor, i. e. $sources(P) = |\mathcal{F}_P|$.

Number of Joins. Joining multiple triple patterns increases the complexity of a query. The number of joins $joins(P)$ is defined for basic graph patterns, i. e. conjunctive joins over a set of triple patterns, as shown below.

Query Selectivity. The proportion between the overall number of triples in the relevant graphs and the number of triples which are actually matched by query patterns is the *query selectivity* $sel(P)$. A query with high selectivity yields less results than a query with low selectivity.

$$joins(P) = \sum_{bgp \in BGP(P)} (|bgp| - 1), \qquad sel(P) = \frac{\sum_{\mathcal{G} \in \mathcal{F}_P} |eval(P, \mathcal{G})|}{\sum_{\mathcal{G} \in \mathcal{F}_P} |\mathcal{G}|}$$

According to the above characteristics, we parameterize benchmark queries such that they cover common queries and corner cases. As result of the query generation we want queries with a specific query structure which span multiple Linked Data sources. Hence, the parameters for join structure and the number of data sources involved are predominant for the query generation. However, there is a dependency between join structure and covered sources. Typical SPARQL queries have path-shaped and star-shaped join patterns or a combination thereof. Path joins span multiple data sources if two patterns are matched by different data sources but have an entity in common which occurs in subject or object position, respectively. Note that for a unique path the number of different data sources is limited by the number of joined triple pattern. Star-shaped join patterns, like {`(?x,isA,foaf:Person)`,`(?x,foaf:name,?name)`}, which match entities in multiple data sources are less interesting for Linked Data queries because they represent unions of unrelated entities.

The combination of path-shaped and star-shaped join patterns produces more complex query structures. They are supported in the query parameterization by defining join rules. These include the join combination, usually via subject or object, and the attachment position with respect to an existing path-shaped join pattern. Corner cases can exhibit a high number of joined triple patterns leading to long paths or "dense" stars. Note that, for reasons of simplicity, we do not consider joins via the predicate position and loops in the query patterns.

5 Query Generation with SPLODGE

With the definition of the query parameterization we can now go into detail of our query generation process SPLODGE. The query generator SPLODGE produces random queries with respect to the query parameters using an iterative approach. In each step, a triple pattern is chosen according to the desired query structure and added if the resulting query pattern fulfills all cardinality constraints. The iteration finishes when all structural constraints are satisfied or when they cannot be satisfied without a violation of the cardinality constraints. In the latter case, the generator cannot produce any query. As the last step, a query is modified with respect to the complexity constraints. In the following, the algorithm and the heuristics are explained in more detail.

5.1 Path Join Pattern Construction

Our query generator SPLODGE starts with the creation of path-shaped join patterns. Let $\mathcal{F} = \{\mathcal{G}_1, \ldots, \mathcal{G}_n\}$ be a federated RDF graph. Given a parameterization of n patterns and m sources ($m \leq n$), the algorithm constructs a sequence of triple pattern

$$PathJoin(n,m) = (t_1, \ldots, t_n) \in ((U \cup L \cup V) \times (U \cup V) \times (U \cup L \cup V))^n$$

such that

$$\forall\, i = 1, \ldots, n-1 : \quad obj(t_i) = subj(t_{i+1}) \text{ and } |\{\mathcal{G}_j \mid \exists j : t_i \in \mathcal{G}_j\}| = m.$$

If $m < n$ then several triple patterns can have the same data source. The distribution of sources among triple patterns is randomly chosen in order to allow for variations.

Computing a valid sequence $PathJoin(n,m)$ directly on the original data, is, in general, infeasible due to the huge search space in a federated RDF graph. We therefore take a heuristic approach which facilities statistical information and cardinality estimation heuristics. In order to limit the effort needed to acquire sophisticated statistics on the various data sources, we restrain our attention to path-shaped join pattern generation with bounded predicates. Note, however, that bound predicates may be replaced with variables in a post-processing step.

Definition 4 (Linked Predicate Patterns). *A linked predicate pattern l is a quadruple $l = (p_1, \mathcal{G}_1, p_2, \mathcal{G}_2)$ with $p_1, p_2 \in U$, $\mathcal{G}_1, \mathcal{G}_2 \in \mathcal{F}$, and $\mathcal{G}_1 \neq \mathcal{G}_2$. The set of valid linked predicate patterns in $\mathcal{F}$ is defined as*

$$\mathcal{L}(\mathcal{F}) = \{(p_1, \mathcal{G}_1, p_2, \mathcal{G}_2) \mid \exists\, s, o, x \in U \cup B \cup L : (s, p_1, x) \in \mathcal{G}_1 \wedge (x, p_2, o) \in \mathcal{G}_2\}$$

Further, we define the following sets of triples which can be matched with the first or second pattern in a linked predicate pattern

$$\phi(p_1, \mathcal{G}_1, p_2, \mathcal{G}_2) = \{(s,x) \mid (s, p_1, x) \in \mathcal{G}_1 \wedge (x, p_2, o) \in \mathcal{G}_2\}$$
$$\tau(p_1, \mathcal{G}_1, p_2, \mathcal{G}_2) = \{(x,o) \mid (s, p_1, x) \in \mathcal{G}_1 \wedge (x, p_2, o) \in \mathcal{G}_2\}$$

The combination of some $\mathcal{L}_1 = (p_1, \mathcal{G}_1, p_2, \mathcal{G}_2)$ and $\mathcal{L}_2 = (p_2, \mathcal{G}_2, p_3, \mathcal{G}_3)$ will only return results if $\tau(p_1, \mathcal{G}_1, p_2, \mathcal{G}_2) \cap \phi(p_2, \mathcal{G}_2, p_3, \mathcal{G}_3) \neq \emptyset$, i. e. bindings for the object o in the first linked predicate patterns must also be part of the join bindings x in the second linked predicate pattern, e. g. {`(?s,dc:creator,?x)`, `(?x,owl:sameAs,?o)`} and {`(?s,owl:sameAs,?y)`,`(?y,dbpp:name,?o)`} can be combined to form a path of three patterns (cf. example 2), but there will be no result if "Mordechai Levin" is bound to $?x$ because he is not included in DBpedia.

In order to compute the result size for a path-shaped join pattern we need to estimate the overlap between two arbitrary linked predicate patterns. A computation of all possible join paths is not feasible for such a large dataset.

Definition 5 (Joined Predicate Pattern Size). *For a sequence of linked predicate patterns $(p_1, \mathcal{G}_1), \ldots, (p_n, \mathcal{G}_n)$ we define the join size as*

$$\prod_{i=1..n} |\sigma_{p_i}(\mathcal{G}_i)| \cdot \prod_{i=2..n-1} js(p_{i-1}, \mathcal{G}_{i-1}, p_i, \mathcal{G}_i, p_{i+1}, \mathcal{G}_{i+1})$$

with $\sigma_{p_i}(\mathcal{G}_i) = \{(s, p_i, o) \in \mathcal{G}_i\}$ *and the join selectivity (js)*

$$js(p_{i-1}, \mathcal{G}_{i-1}, p_i, \mathcal{G}_i, p_{i+1}, \mathcal{G}_{i+1}) = \frac{|\tau(p_{i-1}, \mathcal{G}_{i-1}, p_i, \mathcal{G}_i)| \cdot |\phi(p_i, \mathcal{G}_i, p_{i+1}, \mathcal{G}_{i+1})|}{|\sigma_{p_i}(\mathcal{G}_i)|^2}$$

The pattern join selectivity is a value in the interval $[0, 1]$. The lower the value, the higher the selectivity of the pattern combination and the less results will be returned. In order to prevent pattern combinations that do not return any results, we prefer selectivity values closer to 1.

5.2 Star Join Pattern Construction

Star-shaped join patterns extend path-shaped join pattern at predefined *anchor points*, i. e. at a specific triple pattern in the triple pattern path. Without an anchor point the star join will represent an individual query pattern which is combined via `UNION` with the other query patterns. The query parameterization also defines the number of triple patterns in the star join and if the join variable is in subject or object position. The anchor triple pattern is automatically part of the star join. Hence, it defines the join variable, the source restriction, and the first predicate to be included in the star join.

$$StarJoin(n, \mathcal{G}) = (t_1, \ldots, t_n) \in ((U \cup L \cup V) \times (U \cup V) \times (U \cup L \cup V))^n$$

such that

$$\begin{aligned} \forall\, i = 1, \ldots, n: \quad & subj(t_1) = \ldots = subj(t_n) \;\vee\; obj(t_1) = \ldots = obj(t_n) \;\text{ and} \\ & |\{t \in StarJoin(n, \mathcal{G})\} \cap \{t \in PathJoin(k, l)\}| = 1 \end{aligned}$$

The second condition above formalizes the requirement that the star join intersects with the main path join in one triple pattern. As with path-shaped join pattern, the computation of $StarJoin(n, \mathcal{G})$ combinations on the original data is, in general, infeasible. Thus, statistics-based heuristics are also employed to combine triple patterns with bound predicates. A star-shaped join pattern will only produce results if at least one entity matches all of the triple patterns, i. e. every predicate occurs in a combination with the same entity (always in subject or object position) in the same data source. We utilize *Characteristic Sets* [18] to capture the co-occurrence of predicates with the same entities. Characteristic sets are basically equivalence classes based on distinct predicate combinations. They keep track of the number of different entities and the number for RDF triples for each predicate in the characteristic set. The latter value can be higher due to multi-value predicates. In addition, we extended the statistics with information about the data source a characteristic set occurs in.

Definition 6 (Characteristic Sets). *We define the characteristic set* $S_C(s, \mathcal{G})$ *of a subject* s *(cf. [18]) with respect to a data source* $\mathcal{G}$ *and a federated graph* $\mathcal{F}$ *with* $\mathcal{G} \in \mathcal{F}$ *via*

$$S_C(s, \mathcal{G}) := \{p \,|\, \exists\, o : (s, p, o) \in \mathcal{G}\}$$

and abbreviate $S_C(\mathcal{F}) := \{S_C(s, \mathcal{G}_i) \mid \exists\, s, p, o : (s, p, o) \in \mathcal{G}_i\}$*. Further,* reverse characteristic sets *are used to estimated the result size for star-join patterns with the join variable in object position.*

The number of results for a star-join pattern need to be taken into account for the cardinality estimation of the path-join pattern it is attached to. Therefore, we count the number of triples in source $\mathcal{G}$ which contain the subjects (or objects) of all matching characteristic sets combined with the predicate of the anchor triple pattern. The selectivity is calculated similar to Definition 5 as shown below and multiplied with the cardinality of the path-join pattern:

$$js(\mathcal{G}, (p_1, \ldots, p_n)) := \frac{|\{(s, p_1) \mid \{p_1, \ldots, p_n\} \subseteq S_C(s, \mathcal{G})\}|}{|\sigma_{p_1}(\mathcal{G})|}$$

So far, we described how a star- join pattern is combined with a path-join pattern. This approach is extensible to combine multiple patterns and produce complex queries with mixed join patterns as depicted in Fig. 1. Due to space constraints, we do not go into further details.

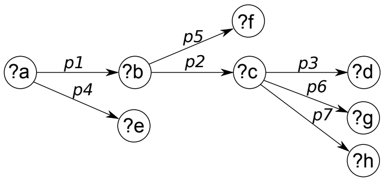


Fig. 1. Query structure generation process. First, triple patterns are combined as path-joins, i. e. ($?a\ p_1\ ?b$), ($?b\ p_2\ ?c$), ($?c\ p_3\ ?d$), then star-joins are created for $?a$, $?b$, $?c$.

6 Implementation

The implementation of the query generation is divided into two phases: statistics collection and the query generation based on the statistics. For our prototype implementation[1] we used the 2011 billion triple challenge dataset[2] containing about 2 billion quads, i. e. subject, predicate, object, and context. We did some pre-processing and cleanup and aggregated all contexts to their common domain name (i. e. {john,jane}.livejournal.com → livejournal.com). As a result, we reduced the 7.4 million different contexts to 789 common domains. Query patterns across different data sources are created based on these reduced domain contexts.

SPLODGE requires statistical information during query generation, i. e. for selecting triple patterns and for estimating the result size and the number of involved data sources. Due to the huge size of the Linked Data cloud, there is a trade-off between the level of statistical details and the overall space requirement for storing the meta data. Therefore, SPLODGE employs only predicate statistics, as the number of distinct predicates is much smaller compared to the number of distinct URIs in the datasets. Moreover, comprehensive statistics also impose a significant processing overhead.

[1] SPLODGE is open source and available at `http://code.google.com/p/splodge/`
[2] `http://km.aifb.kit.edu/projects/btc-2011/`

6.1 Pattern Statistics

For deciding whether triple patterns can be combined during query construction, we need information about the co-occurrence of predicates in RDF statements. For path-joins two predicates co-occur if the respective RDF statements are joined via subject/object. In addition to knowing whether predicates p_1, p_2 co-occur we also need the number $\#_{\mathcal{G}}(p)$ of RDF statements in each $\mathcal{G}$ that mention predicate p, i. e. $\#_{\mathcal{G}}(p) = |\{(s, p, o) \in \mathcal{G}\}|$.

Example 3. Following table shows co-occurrence statistics for path-joins (cf. Def. 4). Each tuple $(p_1, \mathcal{G}_1, n_1, p_2, \mathcal{G}_2, n_2)$ represents a linked predicate pattern $(s, p1, x) \in \mathcal{G}_1 \wedge (x, p2, o) \in \mathcal{G}_2$ with the respective RDF triple counts n_1 and n_2.

p_1	$\mathcal{G}_1$	n_1	p_2	$\mathcal{G}_2$	n_2
owl:sameAs	http://data.gov.uk/	22	foaf:knows	http://dbpedia.org	31
owl:sameAs	http://open.ac.uk/	58	foaf:knows	http://dbpedia.org	17
rdfs:seeAlso	http://bio2rdf.org/	15	rdf:type	http://www.uniprot.org/	38
rdfs:seeAlso	http://zitgist.com/	49	rdfs:label	http://musicbrainz.org/	36

For star-joins, we rely on *Characteristic Sets* [18]. They define equivalence classes for resources based on predicate combinations, i. e. URIs and blank nodes in subject position of RDF statements are in the same characteristic set if they have exactly the same set of predicates. Characteristic sets count the number of entities in such an equivalence class and the number of occurrences for each predicate. The latter helps to identify frequent occurrences of multi-valued predicates. We extend the characteristic sets to include the data source as well.

Example 4. The table below shows the statistical data for a specific characteristic set defined by the predicates (rdf:type, rdfs:label, rdf:sameAs). Each entry is associated with one data source $\mathcal{G}$ and contains the number of resources $\#res$ and the number of RDF triples n_i per predicate p_i in the data source.

$\mathcal{G}$	$\#res$	p_1	n_1	p_2	n_2	p_3	n_3
http://bio2rdf.org/	632	rdf:type	632	rdfs:label	844	owl:sameAs	632
http://www.uniprot.org/	924	rdf:type	924	rdfs:label	924	owl:sameAs	924
http://data.gov.uk/	1173	rdf:type	1421	rdfs:label	1173	owl:sameAs	1399

For the sake of readability, we use URIs (with namespace prefixes) in the examples above. In reality, we employ a dictionary for all predicate and source URIs and only store the entry's index number in the statistics table.

6.2 Verification

The purpose of the verification step is to ensure that all desired constraints are met by the generated queries. The query semantics and the query structure are easy to check by inspecting the syntax and query patterns of the produced SPARQL queries. In fact, these constraints are always met as the query generation is driven by the specified query structure. However, the generated queries may not satisfy the cardinality constraints due to the estimations used by the heuristics. A reliable validation would have to execute each query on the actual data sets which is impossible because of the sheer size of the Linked Data cloud and the absence of a query processing implementation which can execute all possible types of benchmark queries in a short time. Moreover, corner case queries are intended to be hard to evaluate.

Our solution in SPLODGE is to compute a confidence value for each query. It defines how likely a query can return the desired number of results. We reject queries if their confidence value lies below a certain threshold. The confidence value is computed based on the minimum selectivity of all joins in a query.

7 Evaluation

Having explained the technical details and algorithms for the query generation we will now have a look at the evaluation of the generated queries. Evaluation in this context basically means checking if the generated queries meet the predefined cardinality constraints, i. e. if they can actually return results which were obtained from different data sources. Due to the random query generation process using cardinality estimates it is not uncommon that different queries with the same characteristics basically yield different result sizes and cover a range of various data sources. Hence, we want to evaluate two aspects: (1) how many queries in a query set fail to return any result, and (2) how good does the estimated result size match the actual results size of the queries. Furthermore, we count the number of data sources which are involved in answering a queries. To allow for an objective comparison of the effectiveness of the triple pattern selection criteria we perform the query generation with three different heuristics which are used for choosing triple patterns. For this evaluation the queries are restricted to SPARQL SELECT queries with conjunctive joins of triple patterns with bound predicate and unbound subject and unbound object.

7.1 Query Creation Heuristics

A naive approach would just randomly select predicates for use in the triple patterns of the query. For our comparison we use three different pattern selection algorithms, ranging from basic to more elaborate heuristics in order to increase the probability that the generated queries meet the desired constraints.

Baseline uses random selection of data sources and bound predicates in the data sources. It does not check if there is a connection between the data sources via the chosen predicates.

SPLODGE*lite* creates queries as described in Sec. 5. Two triple patterns are combined (in a path-join or star-join) if the statistics indicate the existence of resources which can be matched by the respective predicate combination. SPLODGE*lite* does not apply validation based on a confidence value.

SPLODGE extends SPLODGE*lite* with a computation of confidence values based on individual join selectivity. It discards queries if the confidence value is below a certain threshold, i. e. if individual joins in a query are too selective.

7.2 Setup

Representative query parameterizations are obtained by analyzing the different query structures of the FedBench queries [26]. An overview for the life-science (LS), cross-domain (CD), and linked data (LD) queries is given in Fig. 2. For simplification of the presentation, we only show the join structure and omit bound subjects/objects, filter expressions and optional parts. Queries with unbound predicate were not considered. We can basically model all of the join patterns with the parameterization described in Sec. 4. But due to space restrictions we will only consider selected queries in the evaluation. Query generation and evaluation need to be tailored for a specific dataset. We chose the 2011 billion triple challenge dataset. It contains about two billion triples and covers a large number of Linked Data sources.

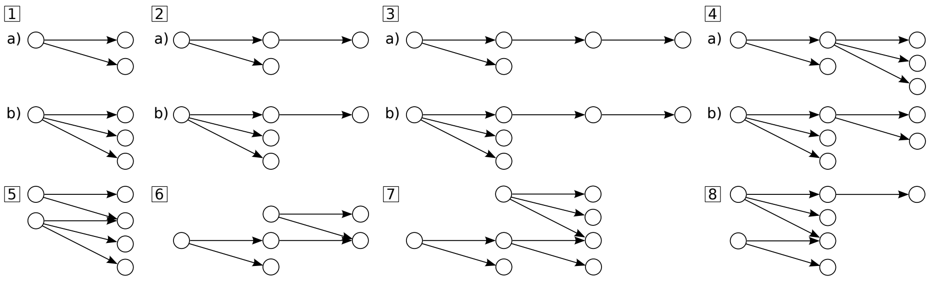


Fig. 2. Query Patterns as exposed by the FedBench queries [26] with bound predicates. Query set (1) has a single star-join, sets (2)-(4) are path-joins combined with star-joins, (5) are two combined star-joins, and sets (6)-(8) are combinations of multiple path-joins and star-joins. (1):LD[5,7], (2):LD[1,2,9,10,11], (3):CD[5,6,7],LS[3],LD[3] (4):CD[3],LD[8], (5):LS[6,7], (6):CD[4], (7):LS[4], (8):LS[5],LD[4].

The major challenge for the query generation is to produce path-join queries across different datasets. The sparsity of links between datasets makes it difficult to create long path-joins and ensure non-empty result sets. To better explore this problem space, we focused in our evaluation on path-join queries where each triple triple pattern needs to be matched by a different data source. Such queries represent interesting corner cases, as all triple patterns must be evaluated independently. Moreover, since the triple patterns have only bound predicates, many data source may be able to return results for a single triple pattern, thus increasing the number of data sources that need to be contacted. We generated sets of 100 random queries for path-joins of length 3–6 and executed them to obtain the actual number of results. All triples of the billion triple challenge dataset 2011 were loaded into a single RDF3X [19] repository.

7.3 Results

The evaluation of all queries from a query set on such a large dataset takes quite long, i. e. several hours for specific queries. The main reason is that some

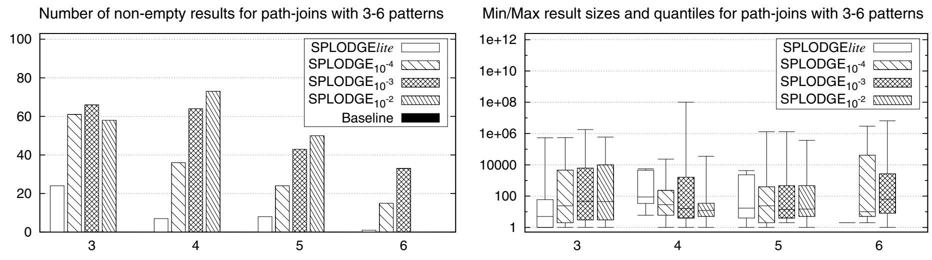


Fig. 3. Comparison of SPLENDID*lite* and SPLENDID using different confidence values, i. e. minimum join selectivity of 0.0001, 0.001, or 0.01, respectively. For each batch of 100 queries the number of non-empty results (left) and the minimum, maximum result size and the quantiles for 0.2, 0.5 and 0.8 (right) are shown.

queries produce very large (intermediate) result sets. A timeout of two minutes is set for a query to finish. For each batch of 100 queries we count the number of queries which returned non-empty results. Additionally, minimum, maximum, median, as well as 0.2 and 0.8 quantiles for all result sizes in a query batch are measured. We compare the results for SPLENDID*lite* and the regular SPLENDID query generation where the confidence value is defined by a minimum selectivity.

Figure 3 shows that SPLENDID*lite* produces only a few queries and the baseline even fails to create any query which can return results. Using the confidence value based on join selectivity increases the number of non-empty results sets significantly, i. e. from 20 to around 60 for three join patterns and by a factor of 3 to 30 for more join patterns. However, for path-joins with six triple patterns it was not possible to generate any query where the minimum join selectivity is 0.01. Considering the minimum and maximum result size we can not see any clear behavior. The minimum result size is always well below 10. The maximum goes in some case up to several million results while the 80% quantile remains below 10000 results (except for a selectivity of 0.001 and six join patterns). All query sets have a median value of less than one hundred results. The difference is smallest when the query sets have a similar number of non-empty results.

We also observe that many predicates in the queries represents schema vocabulary, e. g. `rdf:type`, `rdfs:subClassOf`, `owl:disjointWith`. This becomes even more noticeable the longer the path-join.

8 Summary and Future Work

We presented a methodology and a toolset for systematic benchmarking of federated query processing systems for Linked Data. The novel query generation approach allows for flexible parameterization of realistic benchmark queries which common scenarios and also corner cases. A thorough analysis of query characteristics was conducted to define the dimensions for the parameterization space of queries, including structural, complexity, and cardinality constraints. The implementation of SPLODGE is scalable and has proven to produce useful benchmark queries for the test dataset of the 2011 billion triple challenge.

So far, the query generation handles all predicates equally. With respect to sub-type and sub-property definitions a separate handling of schema information would allow for creating queries suitable for inference benchmarking. For future work, an extension of the statistical information would be helpful to include filter expression in the generated queries. Finally, we intend to use the benchmark queries for the evaluation of federated query processing on Linked Data.

Acknowledgements. The research leading to these results has received funding from the European Community's Seventh Frame Programme under grant agreement No 257859, ROBUST.

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. Alexander, K., Cyganiak, R., Hausenblas, M., Zhao, J.: Describing Linked Datasets – On the Design and Usage of voiD, the Vocabulary Of Interlinked Datasets. In: Proceedings of the Linked Data on the Web Workshop. CEUR (2009)
3. Berners-Lee, T.: Linked Data – Design Issues. Published online (July 27, 2006), http://www.w3.org/DesignIssues/LinkedData.html
4. Bizer, C., Schultz, A.: The Berlin SPARQL Benchmark. International Journal on Semantic Web and Information Systems 5(2), 1–24 (2009)
5. Buil-Aranda, C., Arenas, M., Corcho, O.: Semantics and Optimization of the SPARQL 1.1 Federation Extension. 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. 1–15. Springer, Heidelberg (2011)
6. Duan, S., Kementsietsidis, A., Srinivas, K., Udrea, O.: Apples and Oranges: A Comparison of RDF Benchmarks and Real RDF Datasets. In: Proceedings of the International Conference on Management of Data, pp. 145–156. ACM (2011)
7. Gallego, M.A., Fernández, J.D., Martínez-Prieto, M.A., de la Fuente, P.: An Empirical Study of Real-World SPARQL Queries. In: USEWOD (2011)
8. Görlitz, O., Staab, S.: Federated Data Management and Query Optimization for Linked Open Data. In: Vakali, A., Jain, L.C. (eds.) New Directions in Web Data Management 1. SCI, vol. 331, pp. 109–137. Springer, Heidelberg (2011)
9. Görlitz, O., Staab, S.: SPLENDID: Sparql Endpoint Federation Exploiting Void Descriptions. In: Proc. of the 2nd Int. Workshop on Consuming Linked Data (2011)
10. Guo, Y., Pan, Z., Heflin, J.: LUBM: A Benchmark for OWL Knowledge Base Systems. Web Semantics 3(2-3), 158–182 (2005)
11. Harth, A., Hose, K., Karnstedt, M., Polleres, A., Sattler, K.-U., Umbrich, J.: Data Summaries for On-Demand Queries over Linked Data. In: Proceedings of the 19th International Conference on World Wide Web, pp. 411–420. ACM (2010)
12. 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)

13. Hartig, O., Langegger, A.: A Database Perspective on Consuming Linked Data on the Web. Datenbank-Spektrum 10(2), 57–66 (2010)
14. Hayes, P.: RDF Semantics. W3C Recommendation. Published online (February 10, 2004), http://www.w3.org/TR/2003/PR-rdf-mt-20031215/
15. 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)
16. Möller, K., Hausenblas, M., Cyganiak, R., Grimnes, G.A., Handschuh, S.: Learning from Linked Open Data Usage: Patterns & Metrics. In: Proceedings of the Web Science Conference, pp. 1–8 (2010)
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. Neumann, T., Moerkotte, G.: Characteristic Sets: Accurate Cardinality Estimation for RDF Queries with Multiple Joins. In: 27th International Conference on Data Engineering (ICDE), pp. 984–994 (2011)
19. Neumann, T., Weikum, G.: RDF-3X: a RISC-style Engine for RDF. In: Proceedings of the 34th International Conference on Very Large Data Bases (VLDB), pp. 647–659. VLDB Endowment (2008)
20. Picalausa, F., Vansummeren, S.: What are real SPARQL queries like? In: Proceedings of the International Workshop on Semantic Web Information Management (SWIM), Athens, Greece, pp. 7:1–7:6. ACM (2011)
21. Prud'hommeaux, E., Buil-Aranda, C.: SPARQL 1.1 Federated Query. W3C Working Draft. Published online (November 10, 2011), http://www.w3.org/2009/sparql/docs/fed/service
22. Prud'hommeaux, E., Seaborne, A.: SPARQL Query Language for RDF. W3C Recommendation. Published online (January 15, 2008), http://www.w3.org/TR/rdf-sparql-query/
23. 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)
24. Rodriguez, M.A.: A Graph Analysis of the Linked Data Cloud. Arxiv preprint arXiv:0903.0194, pp. 1–7 (2009)
25. Schenk, S., Staab, S.: Networked Graphs: A Declarative Mechanism for SPARQL Rules, SPARQL Views and RDF Data Integration on the Web. In: Proceedings of the 17th Int'l World Wide Web Conference, Beijing, China, pp. 585–594 (2008)
26. 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)
27. Schmidt, M., Hornung, T., Lausen, G., Pinkel, C.: SP^{2}Bench: A SPARQL Performance Benchmark. In: Proceedings of the 25th International Conference on Data Engineering (ICDE), pp. 222–233 (2009)
28. Schmidt, M., Meier, M., Lausen, G.: Foundations of SPARQL Query Optimization. Arxiv preprint arXiv:0812.3788 (2008)

29. 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)
30. Stuckenschmidt, H., Vdovjak, R., Houben, G.-J., Broekstra, J.: Index Structures and Algorithms for Querying Distributed RDF Repositories. In: Proceedings of the 13th Int'l World Wide Web Conference, New York, USA, pp. 631–639 (2004)
31. Umbrich, J., Hose, K., Karnstedt, M., Harth, A., Polleres, A.: Comparing data summaries for processing live queries over Linked Data. World Wide Web Journal 14(5-6), 495–544 (2011)

RDFS Reasoning on Massively Parallel Hardware

Norman Heino[1] and Jeff Z. Pan[2]

[1] Agile Knowledge Engineering and Semantic Web (AKSW)
Department of Computer Science
Leipzig University, Johannisgasse 26, 04105 Leipzig, Germany
heino@informatik.uni.leipzig.de

[2] Department of Computing Science, University of Aberdeen, Aberdeen AB24 3UE, UK
jeff.z.pan@abdn.ac.uk

Abstract. Recent developments in hardware have shown an increase in parallelism as opposed to clock rates. In order to fully exploit these new avenues of performance improvement, computationally expensive workloads have to be expressed in a way that allows for fine-grained parallelism. In this paper, we address the problem of describing RDFS entailment in such a way. Different from previous work on parallel RDFS reasoning, we assume a shared memory architecture. We analyze the problem of duplicates that naturally occur in RDFS reasoning and develop strategies towards its mitigation, exploiting all levels of our architecture. We implement and evaluate our approach on two real-world datasets and study its performance characteristics on different levels of parallelization. We conclude that RDFS entailment lends itself well to parallelization but can benefit even more from careful optimizations that take into account intricacies of modern parallel hardware.

1 Introduction

Reasoning is an important aspect of the Semantic Web vision. It is used for making explicit previously only implicit knowledge, thus making it available to non-reasoning query engines and as a means for consistency checking during ontology engineering and applications. Applied to large amounts of data, reasoning has been computationally demanding, even if restricted to less expressive RDFS entailment. The idea of concurrent reasoning on parallel hardware has therefore been of research interest for quite some time. Previous work on parallelizing reasoning has focused on cluster-based implementations on top of *shared nothing* architectures that require significant expenses in hardware costs [7,19,21]. Since scalability of parallel RDFS reasoning has been shown by such work, looking at other parallel architectures seems a promising approach.

Over the last years the number of CPU cores available in commodity hardware has increased while the clock rates have not changed much. At the same time graphics processing units (GPUs) have been successfully used for general purpose computing tasks [11]. GPUs provide an even higher level of parallelism by reducing the complexity of a single compute unit and are thus referred to as *massively parallel* hardware. In order to exploit such high levels of parallelism provided by modern hardware, it is essential to devise algorithms in such a way that fine-grained parallelisms become possible.

In this paper, we consider the problem of parallel RDFS reasoning on massively parallel hardware. In contrast to work on cluster-based implementations, we assume a *shared memory* architecture as is found on such hardware and exposed by modern parallel programming frameworks like CUDA[1] or OpenCL[2]. Our goal is to devise and implement an approach that is agnostic of the actual hardware parallelism (i. e. number of cores) and thus able to exploit any degree of parallelism found.

We show that applying the same principles as used in cluster-based approaches in our architecture often incurs a performance impairment, due to massive amounts of duplicates generated by naïve parallel application of RDFS entailment rules. To better understand the nature and origin of those duplicates, we study the problem on two different datasets. We derive two approaches that make use of shared memory in order to prevent duplicate triples from being materialized. Our implementation is based on the OpenCL framework and can thus be used on a wide range of devices, including multicore CPUs and modern GPUs. We evaluate our system on two real-world datasets in the range of tens of millions of triples.

The remainder of this paper is structured as follows. We give an introduction to both classical and parallel RDFS reasoning and discuss related work in Section 2. We describe our approach and its implementation in Section 3 and report on experimental results in Section 4. We conclude and give an outlook on future work in Section 5.

2 Background

2.1 RDFS Entailment

The W3C recommendation on RDF semantics defines a vocabulary with special meaning the interpretation of which can give rise to new triples [5]. In addition, a set of rules are presented whose repeated application is said to yield the RDFS closure of an RDF graph. In this paper, we consider a subset of the RDFS vocabulary that has been shown to capture 'the essence' of RDFS, called the ρdf vocabulary [9]. Table 1 shows those RDFS rules that produce the closure for the ρdf vocabulary. Note that these rules contain all the rules from the RDFS vocabulary that have at least two antecedents. We ignore rules with only one antecedent since, as was already noted in [19], their (trivial) entailments can be computed easily in a single pass over the data. As in previous publications, all figures given in this work have been determined without using RDFS axiomatic triples. However, the system described is capable of including those axiomatic triples that make use of the ρdf vocabulary subset. In particular, the infinite number of axiomatic container membership properties are not included since ρdf does not contain such properties.

2.2 Parallelizing RDFS Entailment

A more detailed inspection of the rules presented in Table 1 reveals two classes. Roughly speaking, there are (i) rules that operate solely on schema triples and (ii) rules that operate on a schema triple and an instance triple.

[1] `http://www.nvidia.com/object/cuda_home_new.html`
[2] `http://www.khronos.org/opencl/`

Table 1. Subset of the RDFS entailment rules with two antecedents

(5)	p `rdfs:subPropertyOf` q	& q `rdfs:subPropertyOf` r	$\Longrightarrow$ p `rdfs:subPropertyOf` r
(11)	C `rdfs:subClassOf` D	& D `rdfs:subClassOf` E	$\Longrightarrow$ C `rdfs:subClassOf` E
(2)	$s\ p\ o$	& p `rdfs:domain` D	$\Longrightarrow$ s `rdf:type` D
(3)	$s\ p\ o$	& p `rdfs:range` R	$\Longrightarrow$ o `rdf:type` R
(7)	$s\ p\ o$	& p `rdfs:subPropertyOf` q	$\Longrightarrow$ $s\ q\ o$
(9)	s `rdf:type` B	& B `rdfs:subClassOf` C	$\Longrightarrow$ s `rdf:type` C

Rules of the first kind compute the *transitive closure* of a property. In Table 1 those rules are shown in the upper section (rules (5) and (11)).

The second kind of rule is shown in the lower part of Table 1 (rules (2), (3), (7), and (9)). We refer to each of these rules as a *join rule*, since it essentially computes a database join between instance and schema triples with the join attribute being the subject of the schema triple and either the property or the object of an instance triple.

Since no rules depend on two instance triples, each can be applied to different instance triples in parallel. Such a *data parallel* task is well suited to GPU workloads due to their ability of efficiently scheduling large numbers of threads. Communication of entailment results is then only necessary between application of different rules. The application of each join rule can thus be considered an *embarrassingly parallel*[3] problem.

```
1 @base <http://example.com/> .
2 @prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
3
4 <p> rdfs:domain <C> .
5 <C> rdfs:subClassOf <D> .
6 <A> <p> "O1", "O2", "O3" .
```

Listing 1. RDF graph that produces duplicates when rules (2) and (9) are applied to it

Treating RDFS reasoning as such a problem, however, can lead to suboptimal performance, since RDFS entailment has an inherent tendency towards producing duplicate triples. To see that, consider the RDF graph in Listing 1. When applying rule (2) to it, each of the triples in line 6 together with the one in line 4 would entail the same triple $\langle A, \texttt{rdf:type}, C\rangle$. Applying rule (9) thereafter would entail the triple $\langle A, \texttt{rdf:type}, D\rangle$, again three times. Since, in this case, duplicates are generated by the same rule we refer to them as *local* duplicates. Another kind of duplicate can be generated by entailing triples that have already been entailed by a previous rule. Those duplicates we refer to as *global* duplicates.

The duplicate problem has been acknowledged in previous work. Urbani et al., for example, combine rules (2) and (3) and cluster instance triples by equality of subject

[3] An embarrassingly parallel problem refers to a problem where very little or no communication is needed.

and object, so as to not entail the same `rdf:type` statements via `rdfs:domain` or `rdfs:range`. The full extent of the problem is not addressed in their work, however, as they do not materialize triples. In Subsection 3.4 we analyze the ramifications that this incurs on performance and discuss approaches to mitigate the problem.

2.3 OpenCL Programming Model

OpenCL is a vendor-agnostic programming model and API for parallel programming. In OpenCL, parallel workloads are expressed in the form of *compute kernels* that are submitted for execution on a parallel device. Since the way in which kernels are written allows for fine-grained parallelism, such devices can range from manycore CPUs to massively parallel GPUs. Execution and submission of kernels are controlled by a host program, which usually runs on the CPU. An instance of a compute kernel (run on a compute device) is called a *work item* or simply a thread[4]. Work items are combined into *work group*s, where those items have access to low-latency shared memory and the ability to synchronize load/store operations using memory barriers. Each work item is assigned a globally (among all work items) and locally (within a work group) unique identifier which also imposes a scheduling order. Those can be used to compute memory offsets for load and store operations. Data transfer between the host system and compute units is done via global memory to which all work items have access, albeit with higher latency. OpenCL provides no means for memory allocation on the device. This means that all output buffers must be allocated on the host *before* executing a kernel. For workloads with a dynamic result size (like RDFS reasoning) this implies a problem. Section 3.3 discusses our approach to this problem.

In this paper we refer to *device* when we talk about the hardware component on which a compute kernel is executed, while we refer to controlling code as being executed on the *host*. Note that in the OpenCL programming model both host and device can refer to the same CPU.

2.4 Related Work

Related work on RDFS entailment has primarily focused on distributing the reasoning workload on a cluster of compute nodes.

An embarrassingly parallel implementation of RDFS entailment over a compute cluster is presented in [21]. Their work uses an iterative fixpoint approach, applying each RDFS rule to triples until no new inferences can be made. Triples generated by different nodes are written to separate files. Thus, duplicates are not detected and consume both memory bandwidth and storage space.

An interesting approach to parallel RDFS reasoning using distributed hash tables is presented by Kaoudi et al. [7]. They give algorithms for both forward and backward chaining and an analytical cost model for querying and storage.

Oren et al. describe a distributed RDFS reasoning system on a peer network [13]. Their entailment is asymptotically complete since only a subset of generated triples are forwarded to other nodes. Duplicate elimination is performed on a node determined by

[4] In this work we use the terms *work item* and *thread* interchangeably.

the hash value of a triple. Each node keeps a local bloom filter with all seen triples and deletes subsequent triples that hash to the same positions. This approach is incompatible with our complete reasoning implementation: bloom filters operate with a controllable false positive rate. A triple that is detected as being a duplicate might actually be a false positive and is thus erroneously deleted.

Urbani et al. describe a parallel implementation of the RDFS rule set using the MapReduce framework [19]. They introduce a topological order of the RDFS rules such that there is no need for fixpoint iteration. Our work in principle is based on this approach by using the same rule ordering (see Section 3.3). However, they do not materialize triples and thus do not encounter the duplicate detection problem.

The most recent work on single-node RDFS entailment is found in [3]. In this work, RDFS entailment rules for annotated data as described in [17] are implemented on top of PostgreSQL. Furthermore, idempotency of rules (5) and (11) is shown. Since their result can easily be extended to classical RDFS entailment (and thus ρdf), we make use of that result as well. In difference to this work, our implementation operates in main memory but has no problem dealing with datasets even bigger than those used by the authors to evaluate their approach.

Rule-based parallel reasoners for the OWL 2 EL profile have been presented for TBox [8] and ABox reasoning [14]. Both works rely on a shared queue for centralized task management. In our work, tasks are described in a much more fine-grained manner and global synchronization is only necessary during kernel passes.

Other related work deals with specialized algorithms for GPU computing. The parallel prefix sum (or scan) over a vector of elements computes for each index i the sum of elements with indexes $0 \ldots i-1$ (exclusive) or $0 \ldots i$ (inclusive). Our scan implementation is based on the work of Sengupta et al. [16], while our sorting algorithm draws inspiration from Satish et al. [15].

3 Approach

In this section we describe our approach and its implementation in detail. We explain how data is prepared and how it is stored. We then describe how each of the ρdf rules is implemented in our system. Subsection 3.4 discusses problems we encountered with regard to duplicate triples and how we address them.

3.1 Data Representation

Parsing and preparing the data are performed in serial which therefore tends to dominate the process. We use the Turtle parser implementation from RDF-3X [10] with a few modifications. All string values (i. e. URIs and literal objects) are encoded to 64-bit integers. This serves three purposes:

- it guarantees that each value is of fixed length, simplifying the implementation of data structures,
- comparing integers is much faster than comparing strings,
- strings are on average longer than 8 bytes which means we essentially perform data compression, thus lowering the required memory bandwidth.

The dictionary is stored in a disk file which is mapped into memory using the `mmap` system call. The file contains buckets storing the literal identifier along with character data and the address of potential overflow buckets. In order to save space, we doubly hash dictionary entries: A hash value is computed for each literal string to be stored in the dictionary. This hash is used as an index into a second hash table that associates it with an offset into the dictionary file. Since literal identifiers are consecutive integers, the reverse index, which maps literal identifiers to string values, is just a vector of file offsets with the id being the vector index (direct addressing). For entries that are less than a virtual memory page size in length, we ensure they do not span a page boundary: If the entry is larger than the space left on the current page (4096 bytes on most systems) we place it entirely on the next page.

We use the 48 least significant bits (i. e. bits 0–47) for value encoding and the 16 most significant bits (bits 48–63) for storing associated metadata (e. g. whether the encoded value is a literal or a blank node). This is needed since RDFS rules as presented in [5] are incomplete if applied to standard RDF graphs [18]. To solve this problem we internally store a generalized RDF graph (i. e. we allow literals in subject position as well as blank nodes in predicate position). For RDF output, however, we need to be able to detect those non-RDF triples.

3.2 Data Storage and Indexing

Data is stored in memory in STL[5] data structures. We use one `std:vector` of terms storing a column of RDF terms which is reminiscent to what column-oriented database systems store [1]. In other words, we keep a separate vector for all subjects, properties, and objects, enabling us to quickly iterate over the attributes of adjacent triples which is a common operation when preparing data for the reasoning stage. In addition, a fourth vector holds flags for each triple; e. g.., entailed triples are stored with an *entailed flag* which allows us to easily determine those.

The index into these vectors is kept in a hash table which is indexed by triple. An amortized constant-time hash lookup is hence sufficient to determine whether a triple has already been stored or not. The cost for this lookup is about half of the cost for actually storing the triple.

The storage layer also implements the iterator concept known from STL containers. A triple iterator can be used to retrieve all stored triples. In order to produce entailed triples only, a special iterator is provided that skips over all triples whose entailed flag is not set.

3.3 Sound and Complete Rule Implementation

As previously noted, our algorithm is based on the rule ordering in [19] with a few differences. Due to each rule being executed in a separate kernel run (pass), our approach involves an implicit *global synchronization* step that is inherent to submitting a kernel to an OpenCL device. Hence all entailments created during one pass are seen by all

[5] Standard Template Library–part of the C++ standard library.

subsequent ones. In particular, if a schema triple is created during a rule application, it will be used as such by subsequent rules. In addition, we perform a fixpoint iteration over rules (5) and (7). This is necessary since rule (7) is able to produce arbitrary triples, including schema triples, and is itself dependent on rule (5). Since it is possible to extend the RDFS vocabulary with custom properties, the fixpoint iteration is necessary to materialize all schema triples before applying other rules. Figure 1 depicts our approach with its four passes as well as synchronization steps in between.

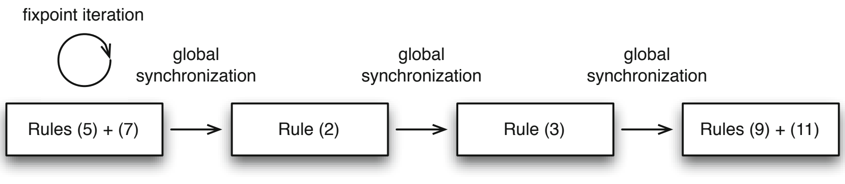


Fig. 1. Passes for computing the subset of the RDFS rules considered in this work

Computing Transitive Closure-Based Rules. Parallel algorithms for computing the transitive closure of a graph are based on boolean matrix multiplication as proposed by Warshall [20]. Due to its regular access pattern, it maps easily to the OpenCL memory model. Transitive property hierarchies on RDF graphs, however, tend to be very sparse. In YAGO2 Core, for instance, the number of vertices taking part in the `rdfs:subClassOf` relation is 365,419, while the number of triples using that property in the full closure is about 3.4 million. Representing such a graph in a quadratic matrix is wasteful since most entries will be zero. In YAGO2 Core, it is also infeasible because storing the adjacency matrix of 365,419 vertices would require almost 16 GiB, if each entry is compressed to a single bit.

A space-efficient serial algorithm for calculating the transitive closure of a graph was presented by Nuutila [12]. We tried the implementation found in the Boost Graph Library[6] but it was unable to cope with the graph size from YAGO2 Core. We thus provide our own implementation of that algorithm which is computed serially on the host. A parallel implementation of the algorithm from [12] is beyond the scope of this paper and will be reserved for future work.

Computing Join Rules. In our parallel implementation, each thread is assigned a single instance triple based on its global identifier. Thus, a join rule needs to find a matching subject of a schema triple.

A very efficient join algorithm in database systems is known as the hash join [4]. Hash joins are typically used in cases where a large relation must be joined with a smaller one. A hash table is populated with the values of the smaller relation. For each value of the large relation, a simple hash lookup can determine whether there is a match. Once a matching schema subject has been found, the objects of all schema triples for that particular rule can be used to materialize new triples.

[6] `http://www.boost.org/libs/graph/`

A hash-join implementation of the process works as follows. Consider rule (9) from Table 1. The object (B) of a given triple is hashed and matched against all `rdfs:subClassOf` schema triples. If a match is found all of its successors (C) become objects of new triples that need to be written to the output vector.

In the OpenCL computing model, this is, however, impossible. All buffers for storing results must be allocated on the host before kernel invocation. We therefore compute join rules in two passes. During the first pass, each thread performs the join with schema triples as described above. Instead of materializing new triples, it just writes the *number of results* it would produce and the index of the matched schema triple to the output. A prefix sum over the result numbers then yields for each thread the number of results that will be written by threads with a lower id. This value is equal to the index into the global result vector at which the thread can write its result. In a second pass each thread reads the matched schema index and materializes new triples.

The hash table with schema elements is stored in two separate vectors. One vector holds bucket information with an offset into the second vector and the number of overflow entries it contains. Since it has an entry for each calculated hash index it can contain empty values when there is no schema subject that hashes to a given value. The second vector stores the buckets with the schema subject and the number of successors packed into a single 64-bit value, followed by the successors of the schema subject. For hashing we use a modified version of Google CityHash[7] which we tailored to an input length of 8 bytes.

3.4 Avoiding Duplicates

As discussed in Subsection 2.2, simple application of RDFS rules can lead to duplicate triples being generated. To study the nature of these duplicates and where they originate from, we analyzed generated triples for each rule on two datasets. Table 2 shows the amount of new triples as well as duplicates generated by each rule for both data sets. Consider rules (2) and (3) in the DBpedia dataset. Rule (2) produces more than 20 times as many duplicates as it produces unique triples while (3) produces about nine times as many. Rules (11) and (9) combined produce more than 40,000 times as many duplicates as useful triples.

This huge amount of duplicates not only waste memory but also bandwidth when being copied from and to the device. For each duplicate triple there must be determined that it actually is a duplicate which requires a hash table lookup in our implementation. The cost can be even higher if other or no index structures are used. Our storage layer can determine whether a given triple is a duplicate in about half the time it takes to store it. Given the figures above, between five and ten times the amount of work done in the storage layer is thus wasted on detecting duplicates. To eliminate this overhead it is important to detect duplicates as early as possible by avoiding their materialization. In the following sections we devise two strategies for dealing with duplicates of different origin as discussed in Subsection 2.2. We refer to these as the G strategy for global duplicates and the L strategy for local duplicates. In Section 4 we compare both strategies with respect to their efficacy on different datasets.

[7] `http://code.google.com/p/cityhash/`

Table 2. Number of triples generated per rule for DBpedia (ca. 26 million triples) and YAGO2 Core (ca. 36 million triples) datasets

	DBpedia			YAGO2 Core		
Rule	Triples	Duplicates	Ratio	Triples	Duplicates	Ratio
(5)	0	0	–	0	19	>
(7)	0	0	–	3,551,361	88,477	0.03
(2)	368,832	7,630,029	21	6,450,781	13,453,038	2.1
(3)	568,715	4,939,870	8.7	409,193	1,511,512	3.7
(11)	259	610	2	3,398,943	366,764	0.1
(9)	0	8,329,278	>	6,685,946	3,173,957	0.5
(11+9)	259	10,398,328	42,162	35,061,599	57,969,000	1.7
all	1,650,607	23,775,152	14	45,766,218	89,370,361	2.0

Preventing Global Duplicates. RDFS rules are verbose—the same conclusions can be derived from different rules. Detecting such duplicates can only be done by allowing each thread a global view of all the triples that are already stored. Since all rules that produce large amounts of duplicates (i. e. (2), (3), and (9)) create triples with `rdf:type` as the predicate, it is sufficient to index only those triples[8]. We use an indexing scheme and a hash table similar to the one used for the schema elements when computing the hash join. Our index is stored in two vectors: one maps the calculated hash value to a bucket address, while the other one holds the buckets. The structure of buckets can be kept simpler since, with a fixed predicate, each needs to contain only a subject and an object. Due to its size, local memory cannot be used and we need to keep the index in *global device memory*. Store operations to global memory cannot be synchronized among threads in different work groups. Thus our index is static and is not extended during computation.

Removing Local Duplicates. Avoiding rule-local duplicates on a global level cannot be done in OpenCL, since global synchronization is not possible during kernel execution. Accordingly, we instead remove those duplicates on the device after they have been materialized but before they are copied to the host. This frees the host from having to deal with those duplicates.

Our procedure for locally removing duplicates is shown in Algorithm 1. It works by first sorting the values in local memory. Each thread then determines whether its neighbor's value is a duplicate of its own value and if so, writes 1 into a flag buffer. A parallel prefix sum is then performed over the flags. Thereafter, the flag buffer contains for each thread the number of duplicate entries in threads with lower ids. If the flag determined by a thread in line 4 was 0 (i. e. its neighbor's value is not a duplicate of its own), it is the first in a series of duplicates. Thus it copies its value to a position in the global output buffer that is k positions lower than its own id, where k is a local displacement value obtained from the scanned flag buffer.

[8] Rule (11), though producing a large number of duplicates, is implemented in a serial algorithm on the host and can thus not be addressed by this strategy.

```
Input: global thread id $g$
Input: local thread id $l$
Input: local buffer lbuf
Input: flag buffer flags
Input: global result buffer gbuf
Result: global buffer gbuf with locally unique values
flags$_l$ $\longleftarrow$ 0, gbuf$_l$ $\longleftarrow$ 0
local_sort(lbuf)
if $l > 0$ then
    if lbuf$_l$ = lbuf$_{l-1}$ then flags$_l$ $\longleftarrow$ 1
    else flags$_l$ $\longleftarrow$ 0
    $f$ $\longleftarrow$ flags$_l$
    prefix_sum(flags)
    if $f = 0$ then
        $k$ $\longleftarrow$ flags$_l$
        gbuf$_{g-k}$ $\longleftarrow$ lbuf$_l$
```

Algorithm 1. Removing duplicates within a work group using local device memory

3.5 Improving Work Efficiency on GPU Devices

On modern GPUs work items do not execute in isolation. Independent of the work group size they are scheduled in groups of 64 (AMD) or 32 (NVIDIA), called *wavefront* or *warp*, respectively. All threads within such a group must execute the same instructions in lock-step. That is, different code paths due to control flow statements are executed by all items, throwing away results that are not needed (predication).

A naïve implementation of our algorithm that loops over all successors of a join match to materialize triples would thus incur wasted bandwidth. To address this problem, we move the loop up to the thread level, effectively increasing the number of work items for the second pass. Each work item then materializes at most one triple. To this end each thread gets passed a local successor index identifying which successor it has to compute for a given subject.

4 Experimental Results

In this section, we evaluate our implementation with two real-world data sets. The DBpedia dataset consists of the DBpedia Ontology, Infobox Types and Infobox Properties from DBpedia 3.7 [2], together amounting to more than 26 million triples. The second dataset is YAGO2 Core [6] in the version released on 2012-01-09, which is sized at about 36 million triples.

We perform two different experiments. First, to show scalability on different levels of hardware parallelism, we measure running times of our reasoning algorithm for each dataset with different numbers of CPU cores used. We achieve this by partitioning the CPU device into sub-devices with the respective number of cores[9]. For this experiment,

[9] This OpenCL 1.2 feature is not yet available on GPUs.

we used an Opteron server with four CPUs having 8 cores each. It is exposed to OpenCL as a single device having 32 compute units (cores). For our experiment we created sub-devices with 16, 8, 4 and 2 compute units.

In order to study effectiveness of our optimizations, we performed another experiment using a GPU device with 20 compute units. We measure kernel running time as well as the time for computing the complete entailment. This includes time for setting up index structures, executing our rule implementations, detecting any duplicates and storing all entailed triples. We give results for no optimization, removing local duplicates, preventing global duplicates as well as for both kinds of optimizations combined.

In order to compare our implementation with existing work we set up the system described by Damásio et al. [3]. We used PostgreSQL 9.1.3 installed on our Ubuntu system and configured it to use 6 GiB of system memory as buffer cache. Time measurements of rule implementations were done by having PostgreSQL print timestamps before and after each experiment and subtracting the values. To set up the system, we followed the authors' blog entry[10]. We performed the largest `rdfs:subClassOf` transitive closure experiment (T2) and the largest full closure experiment (T6) using the non-annotated rule sets and the fastest implementation variant as reported in [3]. Between repetitions we emptied all tables and re-imported the data. The results of this experiment are shown in Table 3. For T6 we had to disable literal detection within the materialization kernel, which normally prevents triples with literal subjects from being materialized. Experiment T2 can be used to determine the baseline speedup that is gained by using a native C++ implementation without GPU acceleration or parallelism over the PL/pgSQL implementation used by Damásio and Ferreira [3]. We determined this baseline speedup to be about 2.6. Experiment T6 is executed about 9.5 times faster by our system. That is, our system actually performs more than three times better than what one could expect given given the baseline speedup.

Table 3. Closure computation times for experiments T2 and T6 done by Damásio and Ferreira [3] repeated on our hardware and the system described in this paper

	Input triples	Output triples	Damásio (ms)	Our system (ms)	Speedup
T2	366,490	3,617,532	23,619.90	9,038.89	2.6 $\times$
T6	1,942,887	4,947,407	18,602.43	1,964.49	9.5 $\times$

Note that experiment T6 has also been done by Urbani et al. [19] on their MapReduce implementation in more than three minutes. For this graph (~1.9 million triples) our system is much faster since the whole graph including index structures fits easily into main memory, while the overhead of the MapReduce framework dominates their experiment. This result would likely change if a significantly larger graph was used. Sufficient disk-based data structures for managing such a graph are, however, beyond the scope of this paper.

[10] `http://ardfsql.blogspot.de/`

4.1 Experimental Setup

Each experiment was repeated five times and the minimum of all five runs was taken. The benchmark system used was a Linux desktop running Ubuntu 12.04 with an Intel Core i7 3770 CPU and 8 GiB of system memory. The GPU used in our experiments was a midrange AMD Radeon HD 7870 with 2 GiB of on-board memory and 20 compute units. For scalability experiments, we used an Opteron server running Ubuntu Linux 10.04. It was equipped with four AMD Opteron 6128 CPUs, each having eight cores and a total 128 GiB of system memory.

All experiments were run using AMD APP SDK version 2.7 and timings were determined using AMD APP profiler version 2.5.1804. C++ code was compiled with Clang 3.1[11] on the desktop and GCC 4.7.1 on the server using `libstdc++` in both cases.

The source code of our implementation is available on GitHub[12]. Due to their size datasets are not part of the source code repository and can be recreated as described in the next section.

4.2 Data Scaling

Both datasets were scaled as follows: (1) all schema triples were separated, (2) instance triples were scaled to 1/2, 1/4th, 1/8th, and 1/16th of the original size, (3) scaled instance triples were combined with all schema triples. The resulting number of instance triples for each dataset are shown in Table 4 along with the number of entailed triples. DBpedia datasets use 3,412 schema triples, YAGO2 datasets contain 367,126 schema triples.

Table 4. Datasets used in our experiments

Dataset	Instance triples	Entailed triples	Dataset	Instance triples	Entailed triples
DBpedia	26,471,572	1,650,607	YAGO2	35,176,410	45,766,218
DBpedia/2	13,235,786	1,266,526	YAGO2/2	17,588,205	41,801,394
DBpedia/4	6,617,893	860,982	YAGO2/4	8,794,102	35,684,268
DBpedia/8	3,308,946	545,002	YAGO2/8	4,397,051	19,045,508
DBpedia/16	1,654,473	318,037	YAGO2/16	2,198,525	10,938,726

4.3 Results and Discussion

Results of our first experiment are depicted in Figure 2. For up to 16 compute units, the kernel running time is approximately halved when the number of cores is doubled. If all 32 compute units are used the running time can be seen to even slightly increase. Note that the Opteron CPUs used in this experiment have a feature similar to Intel's Hyper-threading where some CPU resources are shared by two cores. Thus it remains unclear whether the observed effect is due to limitations of our algorithm or congestion of shared CPU resources.

[11] `http://clang.llvm.org/`
[12] `https://github.com/0xfeedface/grdfs`

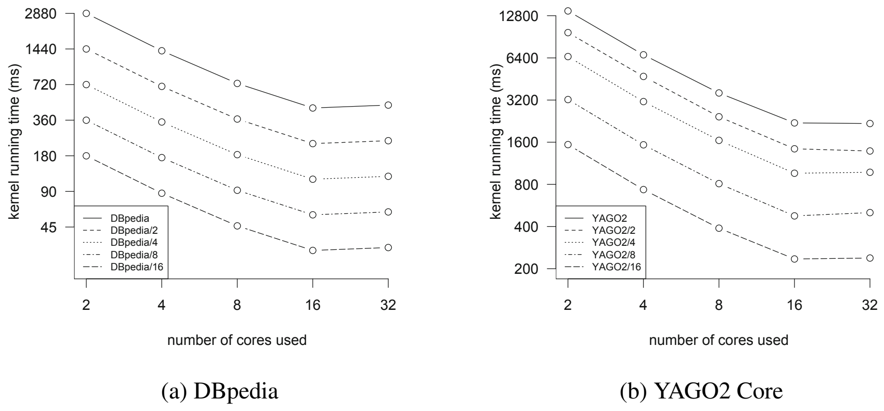


Fig. 2. Kernel running times on different numbers of CPU cores

Table 5 shows benchmark results on the GPU device for two datasets and different combinations of duplicate removal strategies. For DBpedia we use the full dataset, we use the YAGO2/8 dataset only since the full closure of YAGO2 does not fit into the GPU memory. Note that, in this case, the closure would have to be computed in several runs but an algorithm for dynamic data partitioning lies beyond the scope of this paper and will be addressed in future work. In Table 5 one can see that the sorting step involved in the Local strategy does increase the kernel running time to about four to seven times that of the plain kernels without duplicate removal. The result is a reduction of the number of duplicates by factor of up to 13 for YAGO2 and 2 for the DBpedia dataset. The total time needed for computing the closure is reduced by 13 % for DBpedia and 11 % for YAGO2 by this strategy.

The Global deduplication strategy appears to be less effective in terms of overall speedup. Even though the number of duplicates reduced by it for the DBpedia dataset is about seven times that of the Local strategy, closure computation is sped up by only

Table 5. Kernel and complete closure computing times on the GPU device with Local (L), Global (G) or both duplicate removal strategies. The speedup is shown for the complete computation over the *None* strategy.

Dataset	Strategy	Kernel time (ms)	Closure time (ms)	Duplicates	Speedup
DBpedia	None	28.444	6,884.15	23,775,152	
	L	120.915	6,083.76	12,165,520	13.2 %
	G	52.305	6,635.60	1,511,758	3.7 %
	L+G	117.400	6,557.94	1,057,470	5 %
YAGO2/8	None	25.565	21,625.19	31,552,221	
	L	187.169	19,554.09	2,399,898	10.6 %
	G	53.948	21,622.31	29,357,936	0 %
	L+G	215.947	19,807.66	1,786,753	9.2 %

3.7 %. For the YAGO2 dataset, Global deduplication does not lead to a significant decrease in duplicates.

One possible explanation for the reduced efficacy of the Global strategy lies in our implementation. We use a hash table that is built on the host and thus requires a significant amount of serial work. Results shown in Table 5 suggest this cost to be almost half that of removing duplicates. The Local strategy, on the other hand, is performed entirely in parallel, thus coming almost for free when compared to the Global strategy. One possible improvement that we will look at in the future is computing the hash table on the device.

Table 6. Kernel execution and total closure computation time on CPU and GPU

Device	Kernel execution (ms)	Total (ms)
Core i7 3770 (CPU)	647.311	5509.92
Radeon HD 7870 (GPU)	114.683	5881.54

When comparing kernel execution times on GPU and CPU devices (shown in Table 6), one can see that in our system kernels execute about five times faster on the GPU than on the CPU. This is probably due to the large amount of parallelism exposed by modern GPUs. However, this does not translate into shorter closure calculation times. If computation is done on the CPU, host and device memory are essentially the same and no copying takes place. On a GPU however, data must be copied over the PCI Express bus to the device and results have to be copied back. Therefore to fully exploit GPU devices the data transfered must be kept at a minimum. At the moment we do not handle this very well since the OpenCL programming model requires buffers to be allocated in advance. If duplicates are later detected the size of buffers cannot be reduced accordingly. Instead, duplicate values are overwritten with zeros which allows easy detection on the host but does not reduce the amount of data transfered.

5 Conclusion and Future Work

In difference to previous related work that mainly focused on distributed reasoning via a cluster of compute notes, in this paper we tackled the problem of computing a significant subset of the RDFS closure on massively parallel hardware in a shared memory setting. We devised algorithms that can in theory exploit such high levels of parallelism and showed their scalability to at least 16 cores on CPU hardware.

In addition, we addressed two issues that make parallel RDFS reasoning non-trivial: i) we introduced a fixpoint iteration over rules (5) and (7) to support extension of the RDFS vocabulary and ii) we devised two independent strategies for dealing with the large number of duplicate triples that occur in naïve application of RDFS entailment rules. In particular, we could show our Local strategy to be highly effective in terms of speedup gain. The comparatively high cost for the Global strategy make it less effective in our system. However, if the aim lies in maximum reduction of duplicates (e. g. if the cost for duplicate detection is very high) both strategies need to be applied. One aspect of future work is thus reducing the setup cost for the Global strategy by bulding data structures on the device.

The reason for our implementation showing no performance gain on massively parallel GPU hardware over CPU hardware is due to wasted memory bandwidth when copying large amounts of data to and off the device. To overcome this future research on compressed data structures for massively parallel hardware is needed. Currently, we are not aware of any such work.

In order to cope with datasets of arbitrary size our in-memory data store would need to be replaced by a sufficient disk-based storage scheme. Such schemes specifically designed for RDF data and query workloads are already being researched. Since reasoning and query workloads could be different, designing such schemes specifically for reasoning systems might be a promising approach as well.

Massively parallel hardware is also used for cloud computing platforms that combine several devices[13]. So far, our work focused on using a single OpenCL device per reasoning workload. An obvious extension would be the use of several devices by partitioning instance triples and replicating schema triples on all devices.

Lastly, exploiting a different aspect of modern GPU hardware could be the use of annotated RDF data as described in [17]. In this work, each RDF triple is annotated with a real number $\varphi \in [0, 1]$. The tradeoffs typically made in GPU chip design favor floating-point over integer arithmetic. Thus, extended RDFS rules from [17] would likely benefit from being executed on the floating-point units of modern GPUs.

Acknowledgments. The authors would like to thank the anonymous reviewers for their helpful comments on earlier versions of the paper. In particular, we would like to thank Peter F. Patel-Schneider for the fruitful discussion on properties of our approach.

Parts of this work have been funded by the K-Drive project.

References

1. Abadi, D.: Query Execution in Column-Oriented Database Systems. PhD thesis, Massachusetts Institute of Technology (2008)
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. Damásio, C.V., Ferreira, F.: Practical RDF Schema Reasoning with Annotated Semantic Web 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. 746–761. Springer, Heidelberg (2011)
4. DeWitt, D.J., Katz, R.H., Olken, F., Shapiro, L.D., Stonebraker, M.R., Wood, D.A.: Implementation Techniques for Main Memory Database Systems. In: Proc. of the 1984 ACM SIGMOD Intl. Conf. on Management of Data, pp. 1–8. ACM (1984)
5. Hayes, P.: RDF Semantics. W3C Recommendation, W3C (2004), http://www.w3.org/TR/2004/REC-rdf-mt-20040210/

[13] http://aws.amazon.com/about-aws/whats-new/2010/11/15/announcing-cluster-gpu-instances-for-amazon-ec2/

6. Hoffart, J., Berberich, K., Weikum, G.: YAGO2: a Spatially and Temporally Enhanced Knowledge Base from Wikipedia. Artificial Intelligence Journal, Special Issue on Artificial Intelligence, Wikipedia and Semi-Structured Resources (2012)
7. Kaoudi, Z., Miliaraki, I., Koubarakis, M.: RDFS Reasoning and Query Answering on Top of DHTs. In: Sheth, A.P., Staab, S., Dean, M., Paolucci, M., Maynard, D., Finin, T., Thirunarayan, K. (eds.) ISWC 2008. LNCS, vol. 5318, pp. 499–516. Springer, Heidelberg (2008)
8. 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)
9. Muñoz, S., Pérez, J., Gutierrez, C.: Minimal Deductive Systems for RDF. In: Franconi, E., Kifer, M., May, W. (eds.) ESWC 2007. LNCS, vol. 4519, pp. 53–67. Springer, Heidelberg (2007)
10. Neumann, T., Weikum, G.: RDF-3X: a RISC-style engine for RDF. In: Proc. of the VLDB Endowment, pp. 647–659. VLDB Endowment (2008)
11. Nickolls, J., Dally, W.J.: The GPU Computing Era. IEEE Micro 30(2), 56–69 (2010)
12. Nuutila, E.: An efficient transitive closure algorithm for cyclic digraphs. Information Processing Letters 52(4), 207–213 (1994)
13. Oren, E., Kotoulas, S., Anadiotis, G., Siebes, R., Ten Teije, A., van Harmelen, F.: Marvin: A platform for large-scale analysis of Semantic Web data. In: Proc. of the WebSci 2009 (2009)
14. Ren, Y., Pan, J.Z., Lee, K.: Parallel ABox Reasoning of $\mathcal{EL}$ Ontologies. In: Pan, J.Z., Chen, H., Kim, H.-G., Li, J., Wu, Z., Horrocks, I., Mizoguchi, R., Wu, Z. (eds.) JIST 2011. LNCS, vol. 7185, pp. 17–32. Springer, Heidelberg (2012)
15. Satish, N., Harris, M., Garland, M.: Designing Efficient Sorting Algorithms for Manycore GPUs. In: Proc. of the IEEE Intl. Symp. on Parallel & Distributed Processing (2009)
16. Sengupta, S., Harris, M., Zhang, Y., Owens, J.: Scan primitives for GPU computing. In: Proc. of the 22nd ACM SIGGRAPH/EUROGRAPHICS Symposium on Graphics Hardware, pp. 97–106. Eurographics Association (2007)
17. Straccia, U., Lopes, N., Lukácsy, G., Polleres, A.: A General Framework for Representing and Reasoning with Annotated Semantic Web Data. In: Proc. of the Twenty-Fourth AAAI Conference on Artificial Intelligence (AAAI 2010), pp. 1437–1442. AAAI Press (2010)
18. 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, 79–115 (2005)
19. Urbani, J., Kotoulas, S., Oren, E., van Harmelen, F.: Scalable Distributed Reasoning Using MapReduce. In: Bernstein, A., Karger, D.R., Heath, T., Feigenbaum, L., Maynard, D., Motta, E., Thirunarayan, K. (eds.) ISWC 2009. LNCS, vol. 5823, pp. 634–649. Springer, Heidelberg (2009)
20. Warshall, S.: A Theorem on Boolean Matrices. Journal of the ACM 9(1), 11–12 (1962)
21. 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)

An Efficient Bit Vector Approach to Semantics-Based Machine Perception in Resource-Constrained Devices

Cory Henson, Krishnaprasad Thirunarayan, and Amit Sheth

Ohio Center of Excellence in Knowledge-Enabled Computing (Kno.e.sis)
Wright State University, Dayton, Ohio, USA
{cory,tkprasad,amit}@knoesis.org

Abstract. The primary challenge of machine perception is to define efficient computational methods to derive high-level knowledge from low-level sensor observation data. Emerging solutions are using ontologies for expressive representation of concepts in the domain of sensing and perception, which enable advanced integration and interpretation of heterogeneous sensor data. The computational complexity of OWL, however, seriously limits its applicability and use within resource-constrained environments, such as mobile devices. To overcome this issue, we employ OWL to formally define the inference tasks needed for machine perception – explanation and discrimination – and then provide efficient algorithms for these tasks, using bit-vector encodings and operations. The applicability of our approach to machine perception is evaluated on a smart-phone mobile device, demonstrating dramatic improvements in both efficiency and scale.

Keywords: Machine Perception, Semantic Sensor Web, Sensor Data, Mobile Device, Resource-Constrained Environments.

1 Introduction

In recent years, we have seen dramatic advances and adoption of sensor technologies to monitor all aspects of our environment; and increasingly, these sensors are embedded within mobile devices. There are currently over 4 billion mobile devices in operation around the world; and an estimated 25% (and growing) of those are *smart* devices[1]. Many of these devices are equipped with sensors, such as cameras, GPS, RFID, and accelerometers. Other types of external sensors are also directly accessible to mobile devices through either physical attachments or wireless communication protocols, such as Bluetooth. Mobile applications that may utilize this sensor data for deriving context and/or situation awareness abound. Consider a mobile device that's capable of communicating with on-body sensors measuring body temperature, heart rate, blood pressure, and galvanic-skin response. The data generated by these sensors may be analyzed to determine a person's health condition and recommend subsequent action. The value of such applications such as these is obvious, yet difficult challenges remain.

[1] `http://www.digitalbuzzblog.com/2011-mobile-statistics-stats-facts-marketing-infographic/`

The act of observation performed by heterogeneous sensors creates an avalanche of data that must be integrated and interpreted in order to provide knowledge of the situation. This process is commonly referred to as perception, and while people have evolved sophisticated mechanisms to efficiently perceive their environment – such as the use of a-priori knowledge of the environment [1-2] – machines continue to struggle with the task. *The primary challenge of machine perception is to define efficient computational methods to derive high-level knowledge from low-level sensor observation data.* From the scenario above, the high-level knowledge of a person's health condition is derived from low-level observation data from on-body sensors.

Emerging solutions to the challenge of machine perception are using ontologies to provide expressive representation of concepts in the domain of sensing and perception, which enable advanced integration and interpretation of heterogeneous sensor data. The W3C Semantic Sensor Network Incubator Group [3] has recently developed the Semantic Sensor Network (SSN) ontology [4-5] that enables expressive representation of sensors, sensor observations, and knowledge of the environment. The SSN ontology is encoded in the Web Ontology Language (OWL) and has begun to achieve broad adoption within the sensors community [6-8]. Such work is leading to a realization of a Semantic Sensor Web [9].

OWL provides an ideal solution for defining an expressive representation and formal semantics of concepts in a domain. As such, the SSN ontology serves as a foundation for our work in defining the semantics of machine perception. And given the ubiquity of mobile devices and the proliferation of sensors capable of communicating with them, mobile devices serve as an appropriate platform for executing machine perception. Despite the popularity of cloud-based solutions, many applications may still require local processing, e.g., for privacy concerns, or the need for independence from network connectivity in critical healthcare applications. *The computational complexity of OWL, however, seriously limits its applicability and use within resource-constrained environments, such as mobile devices* [10].

To overcome this issue, we develop encodings and algorithms for the efficient execution of the inference tasks needed for machine perception: explanation and discrimination. *Explanation* is the task of accounting for sensory observations; often referred to as hypothesis building [2,11]. *Discrimination* is the task of deciding how to narrow down the multitude of explanations through further observation [1,2]. The efficient algorithms devised for explanation and discrimination use bit vector operations, leveraging environmental knowledge encoded within a two-dimensional bit matrix.

To preserve the ability to share and integrate with knowledge on the Web, lifting and lowering mappings between the semantic representations and the bit vector representations are provided. Using these mappings, knowledge of the environment encoded in RDF (and shared on the Web, i.e., as Linked Data) may be utilized by *lowering* the knowledge to a bit matrix representation. On the other hand, knowledge derived by the bit vector algorithms may be shared on the Web (i.e., as Linked Data), by *lifting* to an RDF representation.

The applicability of our approach to machine perception is evaluated on a smart-phone mobile device, demonstrating dramatic improvements in both efficiency and scale. In this paper, we present three novel contributions towards efficient machine perception in resource-constrained environments:

1. Formal definition of two primary inference tasks, in OWL, that are generally applicable to machine perception – explanation and discrimination.
2. Efficient algorithms for these inference tasks, using bit vector operations.
3. Lifting and lowering mappings to enable the translation of knowledge between the high-level semantic representations and low-level bit-vector representations.

Section 2 discusses the application of the SSN ontology for representing sensor observations and a-priori environmental knowledge. Section 3 specifies explanation and discrimination, as an extension to the SSN ontology. The efficient bit vector algorithms, as well as the lifting and lowering mappings, are provided in Section 4. Our approach is evaluated in Section 5, followed by related work in Section 6, and conclusions in Section 7.

2 Semantic Sensor Network Ontology

The Semantic Sensor Network (SSN) ontology [4-5] was developed by the W3C Semantic Sensor Network Incubator Group [3] to serve the needs of the sensors community. This community is currently using it for improved management of sensor data on the Web, involving annotation, integration, publishing, and search [6-8]. The ontology defines concepts for representing sensors, sensor observations, and knowledge of the environment.

The SSN ontology serves as a foundation to formalize the semantics of perception. In particular, the representation of observations and environmental knowledge are employed. An *observation* (`ssn:Observation`) is defined as a situation that describes an observed feature, an observed property, the sensor used, and a value resulting from the observation (note: prefix *ssn* is used to denote concepts from the SSN ontology). A *feature* (`ssn:FeatureOfInterest`; for conciseness, `ssn:Feature` will be used throughout the paper) is an object or event in an environment, and a *property* (`ssn:Property`) is an observable attribute of a feature. For example, in cardiology, elevated blood pressure is a property of the feature Hyperthyroidism. To determine that blood pressure is *elevated* requires some pre-processing; however, this is outside the scope of this work. An observation is related to its observed property through the `ssn:observedProperty` relation.

Knowledge of the environment plays a key role in perception [1-2]. Therefore, the ability to leverage shared knowledge is a key enabler of semantics-based machine perception. In SSN, knowledge of the environment is represented as a relation (`ssn:isPropertyOf`) between a property and a feature. To enable integration with other ontological knowledge on the Web, this environmental knowledge design pattern is aligned with concepts in the DOLCE Ultra Lite ontology[2]. Figure 1a provides a graphical representation of environmental knowledge in SSN, with mappings to DOLCE. An environmental knowledgebase, storing facts about many features and their observable properties, takes the shape of a bipartite graph. (Throughout the paper, *KB* will be used to refer to *environmental knowledgebase*). Figure 1b shows an example KB with concepts from cardiology.

[2] `http://www.loa-cnr.it/ontologies/DUL.owl`

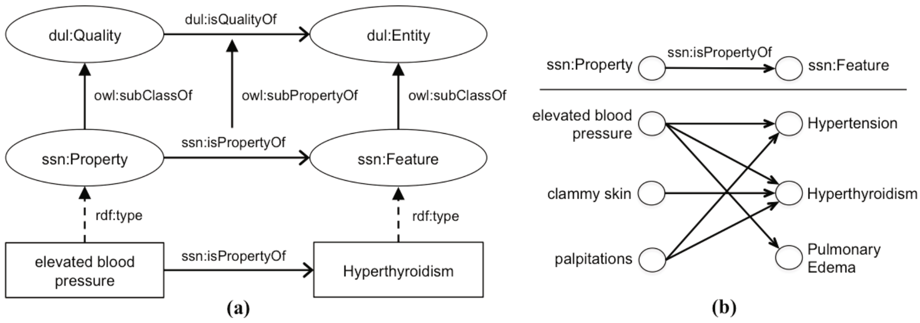


Fig. 1. (a) Graphical representation of environmental knowledge in the SSN ontology, with mappings to DOLCE Ultra Lite (prefix *dul*). **(b)** Graphical representation of an example environmental knowledgebase in cardiology, taking the shape of a bipartite graph. This knowledgebase is derived from collaboration with cardiologists at ezDI (http://www.ezdi.us/).

3 Semantics of Machine Perception

Perception is the act of deriving high-level knowledge from low-level sensory observations [11]. The challenge of machine perception is to define computational methods to achieve this task efficiently. Towards the goal of providing a formal semantics of machine perception, we will define the primary components (inference tasks) of perception in OWL, as an extension of the SSN ontology. The two main components of perception are explanation and discrimination.

3.1 Semantics of Explanation

Explanation is the act of accounting for sensory observations; often referred to as hypothesis building [2,11]. More specifically, explanation takes a set of observed properties as input and yields the set of features that explain the observed properties. A feature is said to *explain* an observed property if the property is related to the feature through an `ssn:isPropertyOf` relation. A feature is said to explain a set of observed properties if the feature explains each property in the set. *Example*: Given the KB in Figure 1b, Hyperthyroidism explains the observed properties elevated blood pressure, clammy skin, and palpitations.

Explanation is used to derive knowledge of the features in an environment from observation of their properties. Since several features may be capable of explaining a given set of observed properties, explanation is most accurately defined as an abductive process (i.e., *inference to the best explanation*) [11]. *Example*: the observed properties, elevated blood pressure and palpitations, are explained by the features Hypertension and Hyperthyroidism (discussed further below). While OWL has not been specifically designed for abductive inference, we will demonstrate that it does provide some of the expressivity needed to derive explanations.

The formalization of explanation in OWL consists of two steps: (1) derive the set of observed properties from a set of observations, and (2) utilize the set of observed properties to derive a set of explanatory features.

ObservedProperty: An *observed property* is a property that has been observed. Note that observations of a property, such as elevated blood pressure, also contain information about the spatiotemporal context, measured value, unit of measure, etc., so the observed properties need to be "extracted" from the observations. To derive the set of observed properties (instances), first create a class `ObservedProperty`. For each observation o in `ssn:Observation` create an existentially quantified property restriction for the `ssn:observedProperty¯` relation, and disjoin them as follows (note: x¯ represents the inverse of relation x):

```
DEF 1: ObservedProperty ≡ ∃ssn:observedProperty¯.{o₁} ⊔ … ⊔
                          ∃ssn:observedProperty¯.{oₙ}
```

ExplanatoryFeature: An *explanatory feature* is a feature that explains the set of observed properties. To derive the set of explanatory features, create a class `ExplantoryFeature`, and for each observed property p in `ObservedProperty` create an existentially quantified property restriction for the `ssn:isPropertyOf¯` relation, and conjoin them as follows:

```
DEF 2: ExplanatoryFeature ≡ ∃ssn:isPropertyOf¯.{p₁} ⊓ … ⊓
                            ∃ssn:isPropertyOf¯.{pₙ}
```

To derive the set of all explanatory features, construct the `ObservedProperty` class and execute the query `ObservedProperty(?x)` with an OWL reasoner. Then, construct the `ExplanatoryFeature` class and execute the query `ExplanatoryFeature(?y)`.

Example: Assume the properties elevated blood pressure and palpitations have been observed, and encoded in RDF (conformant with SSN):

```
ssn:Observation(o1), ssn:observedProperty(o1, elevated blood pressure)
ssn:Observation(o2), ssn:observedProperty(o2, palpitations)
```

Given these observations, the following `ExplanatoryFeature` class is constructed:

```
ExplanatoryFeature ≡ ∃ssn:isPropertyOf¯.{elevated blood pressure} ⊓
                     ∃ssn:isPropertyOf¯.{palpitations}
```

Given the KB in Figure 1b, executing the query `ExplanatoryFeature(?y)` can infer the features, Hypertension and Hyperthyroidism, as explanations:

```
ExplanatoryFeature(Hypertension)
ExplanatoryFeature(Hyperthyroidism)
```

This encoding of explanation in OWL (see DEF 2) provides an accurate simulation of abductive reasoning in the Parsimonious Covering Theory [12], *with the single-feature assumption*[3] [13-14]. The Description Logic expressivity of the explanation task is ALCOI[4,5], with ExpTime-complete complexity [15].

[3] Single-feature assumption specifies that an explanatory feature is a single individual.
[4] Using DL constructs: ⊓, ⊔, ∃, {a}, R¯
[5] `http://www.cs.man.ac.uk/~ezolin/dl/`

3.2 Semantics of Discrimination

Discrimination is the act of deciding how to narrow down the multitude of explanatory features through further observation. The innate human ability to focus attention on aspects of the environment that are essential for effective situation-awareness stems from the act of discrimination [1,2,16]. Discrimination takes a set of features as input and yields a set of properties. A property is said to *discriminate* between a set of features if its presence can reduce the set of explanatory features. *Example*: Given the KB in Figure 1b, the property clammy skin discriminates between the features, Hypertension and Hyperthyroidism (discussed further below).

The ability to identify discriminating properties can significantly improve the efficiency of machine perception [17]. Such knowledge can then be used to task sensors capable of observing those properties.

To formalize discrimination in OWL, we will define three types of properties: *expected property*, *not-applicable property*, and *discriminating property*.

ExpectedProperty: A property is *expected* with respect to (w.r.t.) a set of features if it is a property of every feature in the set. Thus, if it were to be observed, every feature in the set would explain the observed property. *Example*: the property elevated blood pressure is expected w.r.t. the features, Hypertension, Hyperthyroidism, and Pulmonary Edema. To derive the set of expected properties, create a class `ExpectedProperty`, and for each explanatory feature f in `ExplanatoryFeature`, create an existentially quantified property restriction for the `ssn:isPropertyOf` relation, and conjoin them as follows:

DEF 3: `ExpectedProperty` ≡ $\exists$`ssn:isPropertyOf.`$\{f_1\} \sqcap \ldots \sqcap$ $\exists$`ssn:isPropertyOf.`$\{f_n\}$

NotApplicableProperty: A property is *not-applicable* w.r.t. a set of features if it is not a property of any feature in the set. Thus, if it were to be observed, no feature in the set would explain the observed property. *Example*: the property clammy skin is not-applicable w.r.t. the features, Hypertension and Pulmonary Edema. To derive the set of not-applicable properties, create a class `NotApplicableProperty`, and for each explanatory feature f in `ExplanatoryFeature`, create a negated existentially quantified property restriction for the `ssn:isPropertyOf` relation, and conjoin them as follows:

DEF 4: `NotApplicableProperty` ≡ $\neg\exists$`ssn:isPropertyOf.`$\{f_1\} \sqcap \ldots \sqcap$ $\neg\exists$`ssn:isPropertyOf.`$\{f_n\}$

DiscriminatingProperty: A property is *discriminating* w.r.t. a set of features if it is neither expected nor not-applicable. Observing a discriminating property would help to reduce the number of explanatory features. *Example*: As stated above, the property clammy skin is discriminating w.r.t. the features, Hypertension and Hyperthyroidism, as it would be explained by Hyperthyroidism, but not by Hypertension. To derive the set of discriminating properties, create a class, `DiscriminatingProperty`, which is equivalent to the conjunction of the negated `ExpectedProperty` class and the negated `NotApplicableProperty` class.

```
    DEF 5: DiscriminatingProperty ≡ ¬ExpectedProperty ⊓
                                    ¬NotApplicableProperty
```

To derive the set of all discriminating properties, construct the `ExpectedProperty` and `NotApplicableProperty` classes, and execute the query `DiscriminatingProperty(?x)`.

Example: Given the explanatory features from the previous example, Hypertension and Hyperthyroidism (Section 3.1), the following classes are constructed:

```
ExpectedProperty ≡  ∃ssn:isPropertyOf.{Hypertension} ⊓
                    ∃ssn:isPropertyOf.{Hyperthyroidism}

NotApplicableProperty ≡ ¬∃ssn:isPropertyOf.{Hypertension} ⊓
                        ¬∃ssn:isPropertyOf.{Hyperthyroidism}
```

Given the KB in Figure 1b, executing the query `DiscriminatingProperty(?x)` can infer the property clammy skin as discriminating:

```
DiscriminatingProperty(clammy skin)
```

To choose between Hypertension and Hyperthyroidism, task a sensor to measure galvanic skin response (i.e., for clammy skin). The Description Logic expressivity of the discrimination task is ALCO[6], with PSpace-complete complexity [15].

4 Efficient Bit Vector Algorithms for Machine Perception

To enable their use on resource-constrained devices, we now describe algorithms for efficient inference of explanation and discrimination. These algorithms use bit vector encodings and operations, leveraging a-priori knowledge of the environment. Note that this work does not support reasoning for all of OWL, but supports what is needed for machine perception, which is useful in a variety of applications. Table 1 summarizes the data structures used by our algorithms.

Table 1. Quick summary of data structures used by the bit vector algorithms

(note: |x| represents the number of members of x).

Name	Description	About (type, size)
KB_{BM}	Environmental knowledge	Bit matrix of size |ssn:Property| x |ssn:Feature|
$OBSV_{BV}$	Observed properties	Bit vector of size |ssn:Property|
$EXPL_{BV}$	Explanatory features	Bit vector of size |ssn:Feature|
$DISC_{BV}$	Discriminating properties	Bit vector of size |ssn:Property|

4.1 Lifting and Lowering of Semantic Data

To preserve the ability to share and integrate with knowledge on the Web, lifting and lowering mappings between the semantic representations and bit vector representations

[6] Using DL constructs: ⊓, ∃, {a}, ¬C.

are provided. Using these mappings, knowledge of the environment encoded in RDF, as well as observed properties encoded in RDF, may be utilized by *lowering* them to a bit vector representation. Knowledge derived by the bit vector algorithms, including observed properties, explanatory features, and discriminating properties, may be shared on the Web, by *lifting* them to an RDF representation.

Environmental knowledge: An environmental knowledgebase is represented as a bit matrix KB_{BM}, with rows representing properties and columns representing features. $KB_{BM}[i][j]$ is set to 1 (true) iff the property p_i is a property of feature f_j. To *lower* an SSN KB encoded in RDF: for all properties p_i in `ssn:Property`, create a corresponding row in KB_{BM}, and for all features f_j in `ssn:Feature`, create a corresponding column. Set $KB_{BM}[i][j]$ to 1 iff there exists a `ssn:isPropertyOf(`p_i`,`f_j`)` relation. Figure 2a shows an example KB, from Figure 1b, which has been lowered to a bit matrix representation. Index tables are also created to map between the URI's for concepts in the semantic representation to their corresponding index positions in the bit vector representation. Figures 2b and 2c show example index tables for properties and features.

	Hypertension	Hyperthyroidism	Pulmonary Edema
elevated blood pressure	1	1	1
clammy skin	1	0	0
palpitations	1	1	0

(a)

index position	URI
0	http://example.com/cardio.owl#elevated-blood-pressure
1	http://example.com/cardio.owl#clammy-skin
2	http://example.com/cardio.owl#palpitations

(b)

index position	URI
0	http://example.com/cardio.owl#Hypertension
1	http://example.com/cardio.owl#Hyperthyroidism
2	http://example.com/cardio.owl#Pulmonary-Edema

(c)

Fig. 2. (a) Example environmental knowledgebase in the domain of cardiology, from Figure 1b, represented as a bit matrix. Index tables are used for lifting and lowering environmental knowledge between a semantic representation and bit vector representation. (b) Index table for properties. **(c)** Index table for features.

Observed Properties: Observed properties are represented as a bit vector $OBSV_{BV}$, where $OBSV_{BV}[i]$ is set to 1 iff property p_i has been observed. To *lower* observed properties encoded in RDF: for each property p_i in `ssn:Property`, $OBSV_{BV}[i]$ is set to 1 iff `ObservedProperty(`p_i`)`. To *lift* observed properties encoded in $OBSV_{BV}$: for each index position i in $OBSV_{BV}$, assert `ObservedProperty(`p_i`)` iff $OBSV_{BV}[i]$ is set to 1. To generate a corresponding observation o, create an individual o of type `ssn:Observation`, `ssn:Observation(o)`, and assert `ssn:observedProperty(o,`p_i`)`.

Explanatory Features: Explanatory features are represented as a bit vector $EXPL_{BV}$. $EXPL_{BV}[j]$ is set to 1 iff the feature f_j explains the set of observed properties represented in $OBSV_{BV}$ (that is, it explains all properties in $OBSV_{BV}$ that are set to 1). To *lift* explanatory features encoded in $EXPL_{BV}$: for each index position j in $EXPL_{BV}$, assert `ExplanatoryFeature(`f_j`)` iff $EXPL_{BV}[j]$ is set to 1.

Discriminating Properties: Discriminating properties are represented as a bit vector $DISC_{BV}$ where $DISC_{BV}[i]$ is set to 1 iff the property p_i discriminates between the set of explanatory features represented in $EXPL_{BV}$. To *lift* discriminating properties encoded in $DISC_{BV}$: for each index position i in $DISC_{BV}$, assert `DiscriminatingProperty(`p_i`)` iff $DISC_{BV}[i]$ is set to 1.

4.2 Efficient Bit Vector Algorithm for Explanation

Algorithm 1: Explanation

```
[1] input OBSV_BV
[2] define BitVector EXPL_BV
[3] for each j := 0 … |ssn:Feature|-1
[4]    EXPL_BV[j] := 1
[5] for each i := 0 … |ssn:Property|-1
[6]    if OBSV_BV[i] = 1 then
[7]       EXPL_BV := EXPL_BV AND (row i in KB_BM)
[8] output EXPL_BV
```

The strategy employed for efficient implementation of the explanation task relies on the use of the bit vector AND operation to discover and *dismiss* those features that cannot explain the set of observed properties. It begins by considering all the features as potentially explanatory, and iteratively dismisses those features that cannot explain an observed property, eventually converging to the set of all explanatory features that can account for all the observed properties. Note that the input $OBSV_{BV}$ can be set either directly by the system collecting the sensor data or by translating observed properties encoded in RDF (as seen in Section 4.1).

We will now sketch the correctness of the explanation algorithm w.r.t. the OWL specification (Section 3.1). For each index position in $EXPL_{BV}$ that is set to 1, the corresponding feature explains all the observed properties. (*See note about indices*[7]).

Theorem 1: Given an environmental knowledgebase KB, and it's encoding as described in Section 4.1 (i.e., KB_{BM}), the following two statements are equivalent:

S1: The set of *m* observed properties $\{p_{k1}, …, p_{km}\}$, i.e., `ObservedProperty(`p_{k1}`)` ⊓ … ⊓ `ObservedProperty(`p_{km}`)`, is explained by the feature f_e, implies `ExplanatoryFeature(`f_e`)`.

S2: The Hoare triple[8] holds: $\{ \forall i \in \{1, …, m\}: OBSV_{BV}[ki] = 1 \}$
Algorithm 1: Explanation
$\{ EXPL_{BV}[e] = 1 \}$.

Proof (*S1* ⇒ *S2*): The `ObservedProperty` assertions are captured by the proper initialization of $OBSV_{BV}$, as stated in the precondition. Given (i) *S1*, (ii) the single-feature assumption, (iii) the definition: `ExplanatoryFeature ≡ ∃ssn:isPropertyOf⁻.{`p_{k1}`} ⊓ … ⊓ ∃ssn:isPropertyOf⁻.{`p_{km}`}`, and (iv) the fact that `ExplanatoryFeature(`f_e`)` is provable, it follows that $\forall i \in \{1, …, m\}$: `ssn:isPropertyOf(`$p_{ki}$`,`$f_e$`)` is in KB. By our encoding, $\forall i \in \{1, …, m\}$: $KB_{BM}[ki][e] = 1$. Using lines 5-7, the fact that $EXPL_{BV}[e]$ is initialized to

[7] Note that property p_{ki} has property index ki and feature f_{ej} has feature index ej. So ki ranges over 0 to |ssn:Property|-1 and e/ej range over 0 to |ssn:Feature|-1. i and j are merely indices into the enumeration of observed properties and their explanatory features, respectively. Thus, i ranges over 1 to |ssn:Property| and j ranges over 1 to |ssn:Feature|. (In practice, initially i is small and j is large, and through each cycle of explanation and discrimination, i increases while j diminishes.)

[8] {P} S {Q} where P is the pre-condition, S is the program, and Q is the post-condition.

1 and is updated only for i ∊ {1, …, m} where $OBSV_{BV}[ki] = 1$, we get the final value of $EXPL_{BV}[e] = KB_{BM}[k1][e]$ AND … AND $KB_{BM}[km][e] = 1$ (true).

(***S2*** ⇒ ***S1***)**:** Given that {∀i ∊ {1, …, m}: $OBSV_{BV}[ki] = 1$} and {$EXPL_{BV}[e] = 1$} (pre and post conditions), it follows that ∀i ∊ {1, …, m}: $KB_{BM}[ki][e] = 1$ must hold. According to our encoding, this requires that ∀i ∊ {1, …, m}: `ssn:isPropertyOf(`p_{ki}`,e)` holds. Using the definition of `ExplanatoryFeature`, it follows that `ExplanatoryFeature(e)` is derivable (that is, f_e explains *all* the observed properties {p_{k1}, …, p_{km}}).

Theorem 2: The explanation algorithm (Algorithm 1) computes all and only those features that can explain all the observed properties.

Proof: The result follows by applying Theorem 1 to all explanatory features. *Q.E.D.*

4.3 Efficient Bit Vector Algorithm for Discrimination

The strategy employed for efficient implementation of the discrimination task relies on the use of the bit vector AND operation to discover and indirectly *assemble* those properties that discriminate between a set of explanatory features. The discriminating properties are those that are determined to be neither expected nor not-applicable.

Algorithm 2: Discrimination

```
input $EXPL_{BV}$, $OBSV_{BV}$
define BitVector $DISC_{BV}$
for each i := 0 … |ssn:Property|-1
   $DISC_{BV}$[i] := 0
define BitVector $ZERO_{BV}$
for each j := 0 … |ssn:Feature|-1
   $ZERO_{BV}$[j] := 0
for each i := 0 … |$OBSV_{BV}$|-1
   if $OBSV_{BV}$[i] = 0 then
      BitVector $PEXPL_{BV}$ :=
         $EXPL_{BV}$ AND (row i in $KB_{BM}$)
      if $PEXPL_{BV}$ != $EXPL_{BV}$ and
         $PEXPL_{BV}$ != $ZERO_{BV}$ then
         $DISC_{BV}$[i] := 1
output $DISC_{BV}$
```

In the discrimination algorithm, both the discriminating properties bit vector $DISC_{BV}$ and the zero bit vector $ZERO_{BV}$, are initialized to zero. For a not-yet-observed property at index ki, the bit vector $PEXPL_{BV}$ can represent one of three situations: (i) $PEXPL_{BV} = EXPL_{BV}$ holds and the ki^{th} property is expected; (ii) $PEXPL_{BV} = ZERO_{BV}$ holds and the ki^{th} property is not-applicable; or (iii) the ki^{th} property discriminates between the explanatory features (and partitions the set). Eventually, $DISC_{BV}$ represents all those properties that are *each* capable of partitioning the set of explanatory features in $EXPL_{BV}$. Thus, observing any one of these will narrow down the set of explanatory features.

We will now sketch the correctness of the discrimination algorithm w.r.t. the OWL specification (Section 3.2). Each explanatory feature explains all the observed properties. Lemma 1 shows that this is equivalent to all the observed properties being expected properties of the explanatory features.

Lemma 1: If *m* observed properties {p_{k1}, …, p_{km}}, i.e., `ObservedProperty(`p_{k1}`)` ⊓ … ⊓ `ObservedProperty(`p_{km}`)`, are explained by *n* features {f_{e1}, …, f_{en}}, i.e., `ExplanatoryFeature(`f_{e1}`)` ⊓ … ⊓ `ExplanatoryFeature(`f_{en}`)`, then the following holds: ∀i: $1 \leq i \leq m$: `ObservedProperty(`p_{ki}`)` ⇒ `ExpectedProperty(`p_{ki}`)`.

Proof Sketch: The result is obvious from the definition: `ExplanatoryFeature ≡ ∃ssn:isPropertyOf¯.{`p_{k1}`}` ⊓ … ⊓ `∃ssn:isPropertyOf¯.{`p_{km}`}`. So, ∀i, ∀j: $1 \leq i \leq m \wedge 1 \leq j \leq n$: `ssn:isPropertyOf(`$p_{ki}$`,`$f_{ej}$`).ExpectedProperty ≡ ∃ssn:isPropertyOf.{`$f_{e1}$`}` ⊓ … ⊓ `∃ssn:isPropertyOf.{`$f_{en}$`}`.

Lemma 2 restates the assertion (from Lemma 1) that observed properties are also expected properties of explanatory features, in terms of the bit vector encoding.

Lemma 2: The initial values of $EXPL_{BV}$ and $OBSV_{BV}$ satisfy the assertion: ∀ki: ($OBSV_{BV}$[ki] = 1) ⇒ [∀e: ($EXPL_{BV}$[e] = 1) ⇒ (KB_{BM}[ki][e]) = 1)]. And hence, ∀i: ($OBSV_{BV}$[ki] = 1) ⇒ [∀e: ($EXPL_{BV}$[e] ∧ KB_{BM}[ki][e]) = $EXPL_{BV}$[e])].

Proof Sketch: The claim follows from Lemma 1 and the bit vector encoding.

Lemma 3 generalizes Lemma 2 to elucidate an efficient means to determine when a not-yet-observed property is expected, w.r.t. a set of explanatory features.

Lemma 3: Given property ki (p_{ki}) has not-yet been observed, i.e., $OBSV_{BV}$[ki] = 0, `ExpectedProperty(`p_{ki}`)` iff ∀e: ($EXPL_{BV}$[e] ∧ KB_{BM}[ki][e]) = $EXPL_{BV}$[e].

Lemma 4 demonstrates an efficient means to determine when a not-yet-observed property is not-applicable, w.r.t. a set of explanatory features.

Lemma 4: For explanatory features $EXPL_{BV}$ {f_e | $EXPL_{BV}$[e] = 1}, `NotApplicableProperty(`p_{ki}`)` iff ∀e: ($EXPL_{BV}$[e] ∧ KB_{BM}[ki][e]) = $ZERO_{BV}$[e].

Proof Sketch: The result follows from: (i) the definition of `NotApplicableProperty` w.r.t. the set of explanatory features: `NotApplicableProperty(`p_{ki}`)` iff ∀ki, ∀e: `ExplanatoryFeature(`f_e`)` ⇒ `¬∃ssn:isPropertyOf(`p_{ki}`,`f_e`)`; (ii) [∀e: `ExplanatoryFeature(`f_e`)` iff $EXPL_{BV}$[e] = 1]; and (iii) ∀ki, ∀e: [`¬∃ssn:isPropertyOf(`p_{ki}`,`f_e`)` ⇒ KB_{BM}[ki][e] = 0].

Theorem 3: The discrimination algorithm (Algorithm 2) computes all and only those properties that can discriminate between the explanatory features.

Proof: A not-yet-observed property is *discriminating* if it is neither expected nor not-applicable. The result follows from the definition of discriminating property, Lemma 3, and Lemma 4. *Q.E.D.*

5 Evaluation

To evaluate our approach, we compare two implementations of the explanation and discrimination inference tasks. The first utilizes an OWL reasoner as described in Section 3, and the second utilizes the bit vector algorithms described in Section 4. Both implementations are coded in Java, compiled to a Dalvik[9] executable, and run on a Dalvik virtual machine within Google's Android[10] operating system for mobile

[9] `http://code.google.com/p/dalvik/`

[10] `http://www.android.com/`

devices. The OWL implementation uses Androjena[11], a port of the Jena Semantic Web Framework for Android OS. The mobile device used during the evaluation is a Samsung Infuse[12], with a 1.2 GHz processor, 16GB storage capacity, 512MB of internal memory, and running version 2.3.6 of the Android OS.

To test the efficiency of the two approaches, we timed and averaged 10 executions of each inference task. To test the scalability, we varied the size of the KB along two dimensions – varying the number of properties and features. In the OWL approach, as the number of observed properties increase, the `ExplanatoryFeature` class (DEF 2) grows more complex (with more conjoined clauses in the complex class definition). As the number of features increase, the `ExpectedProperty` class (DEF 3) and `NotApplicableProperty` class (DEF 4) grows more complex. In the bit vector approach, as the number of properties increase, the number of rows in KB_{BM} grows. As the number of features increase, the number of columns grows.

To evaluate worst-case complexity, the set of relations between properties and features in the KB form a *complete bi-partite graph*[13]. In addition, for the explanation evaluations, every property is initialized as an observed property; for the discrimination evaluations, every feature is initialized as an explanatory feature. This creates the worst-case scenario in which every feature is capable of explaining every property, every property needs to be explained, and every feature needs to be discriminated between. The results of this evaluation are shown in Figure 3.

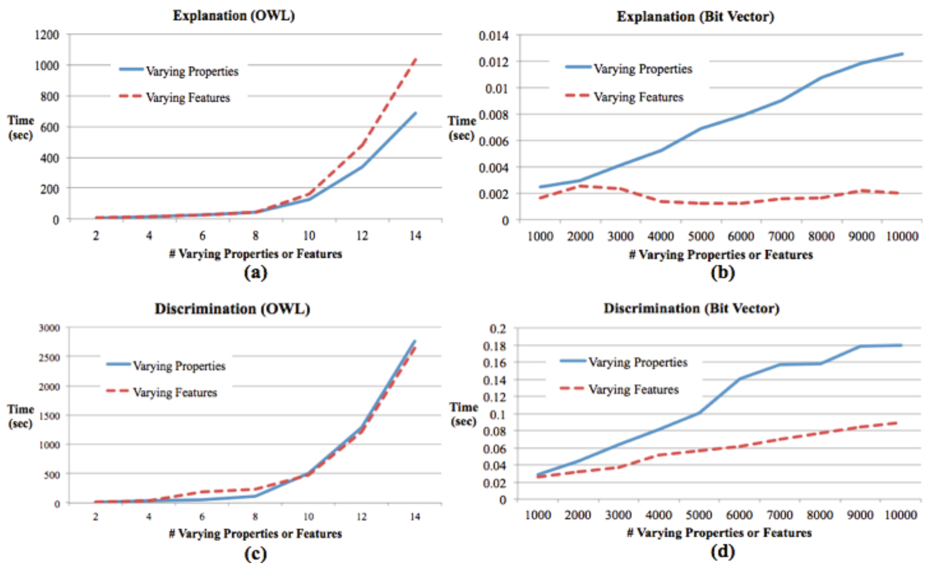


Fig. 3. Evaluation results: **(a)** Explanation (OWL) with $O(n^3)$ growth, **(b)** Explanation (bit vector) with O(n) growth, **(c)** Discrimination (OWL) with $O(n^3)$ growth, and **(d)** Discrimination (bit vector) with O(n) growth.

[11] `http://code.google.com/p/androjena/`

[12] `http://www.samsung.com/us/mobile/cell-phones/SGH-I997ZKAATT`

[13] `http://en.wikipedia.org/wiki/Complete_bipartite_graph` (accessed: June 8, 2012).

Result of OWL Evaluations: The results from the OWL implementations of explanation and discrimination are shown in Figures 3a and 3c, respectively. With a KB of 14 properties and 5 features, and 14 observed properties to be explained, explanation took 688.58 seconds to complete (11.48 min); discrimination took 2758.07 seconds (45.97 min). With 5 properties and 14 features, and 5 observed properties, explanation took 1036.23 seconds to complete (17.27 min); discrimination took 2643.53 seconds (44.06 min). In each of these experiments, the mobile device runs out of memory if the number of properties or features exceeds 14. The results of varying both properties and features show greater than cubic growth-rate ($O(n^3)$ or worse). For explanation, the effect of features dominates; for discrimination, we are unable to discern any significant difference in computation time between an increase in the number of properties vs. features.

Result of Bit Vector Evaluations: The results from the bit vector implementations of explanation and discrimination are shown in Figures 3b and 3d, respectively. With a KB of 10,000 properties and 1,000 features, and 10,000 observed properties to be explained, explanation took 0.0125 seconds to complete; discrimination took 0.1796 seconds. With 1,000 properties and 10,000 features, and 1,000 observed properties, explanation took 0.002 seconds to complete; discrimination took 0.0898 seconds. The results of varying both properties and features show linear growth-rate ($O(n)$); and the effect of properties dominates.

Discussion of Results: The evaluation demonstrates orders of magnitude improvement in both efficiency and scalability. The inference tasks implemented using an OWL reasoner both show greater than cubic growth-rate ($O(n^3)$ or worse), and take many minutes to complete with a small number of observed properties (up to 14) and small KB (up to 19 concepts; #properties + #features). While we acknowledge the possibility that Androjena may have shortcomings (such as an inefficient reasoner and obligation to compute all consequences), our results are in line with Ali et al. [10] that also found OWL inference on resource-constrained devices to be infeasible. On the other hand, the bit vector implementations show linear growth-rate ($O(n)$), and take milliseconds to complete with a large number of observed properties (up to 10,000) and large KB (up to 11,000 concepts).

Consider the mobile application in which a person's health condition is derived from on-body sensors. A person's condition must be determined quickly, i.e., within seconds (at the maximum), so that decisive steps can be taken when a serious health problem is detected. Also, for the application to detect a wide range of disorders (i.e., features) from a wide range of observed symptoms (i.e., properties) the KB should be of adequate size and scope. In practice, an application may not require a KB of 11,000 concepts; however, many applications would require more than 19 concepts.

The comparison between the two approaches is dramatic, showing asymptotic order of magnitude improvement; with running times reduced from minutes to milliseconds, and problem size increased from 10's to 1000's. For the explanation and discrimination inference tasks executed on a resource-constrained mobile device, the evaluation highlights both the limitations of OWL reasoning and the efficacy of specialized algorithms utilizing bit vector operations.

6 Related Work

The ability to derive high-level knowledge from low-level observation data is a challenging task. As argued in this paper, a promising approach to machine perception involves the use of Semantic Web technologies. This approach is quickly evolving into an active area of research. Our work differs from related efforts in three ways: (1) the use of OWL for defining the perception inference tasks, (2) the definition of perception as an abductive process, and (3) the efficient execution of the inference tasks using bit vector operations.

Previous works have utilized OWL for representing concepts in the domain of sensing [4,5,18,19]. Subsequently, First-Order Logic (FOL) rules were often employed to derive knowledge of the features in the environment [20-22]. Taylor et al. [23] have used Complex Event Processing to derive knowledge of events from observation data encoded in SSN. However, as we have shown, several inference tasks useful for machine perception do not require the full expressivity of FOL; they are expressible in OWL, a decidable fragment of FOL.

Second, as opposed to approaches using deductive (FOL) rules, we believe that perception is an abductive process [11]. The integration of OWL with abductive reasoning has been explored [24]; requiring modification of OWL syntax and/or inference engine [25]. We demonstrated that, under the single-feature assumption, abductive consequences can be computed using standard OWL reasoners.

And third, while OWL is decidable, the computational complexity still limits its practical use within resource-constrained environments. A recent W3C Member Submission [26] proposes a general-purpose RDF binary format for efficient representation, exchange, and query of semantic data; however, OWL inference is not supported. Several approaches to implementing OWL inference on resource-constrained devices include [10,27,28,29]. Preuveneers et al. [28] have presented a compact ontology encoding scheme using prime numbers that is capable of class-subsumption. Ali et al. [10] have developed Micro-OWL, an inference engine for resource-constrained devices implementing a subset of OWL constructs, but it is not expressive enough for our inference tasks. McGlothlin et al. [30] serialize RDF datasets and materialize data inferred through OWL reasoning using bit vectors. For our inference tasks, however, it is not scalable. Since we cannot predict which observed properties require explanation, this approach would generate and materialize an `ExplanatoryFeature` class for all possible (exponentially many) combinations of observable properties. In contrast, we have deployed specially tailored linear algorithms that compute explanation and discrimination efficiently.

7 Conclusions and Future Work

We have demonstrated an approach to machine perception on resource-constrained devices that is simple, effective, and scalable. In particular, we presented three novel contributions: (1) a simple declarative specification (in OWL) of two inference tasks useful for machine perception, explanation and discrimination; (2) efficient algorithms for these inference tasks, using bit vector operations; and (3) lifting and lowering mappings to enable the translation of knowledge between semantic representations and the bit vector representations.

The bit vector encodings and algorithms yield significant and necessary computational enhancements – including asymptotic order of magnitude improvement, with running times reduced from minutes to milliseconds, and problem size increased from 10's to 1000's. The approach is prototyped and evaluated on a mobile device, with promising applications of contemporary relevance (e.g., healthcare/cardiology). Currently, we are collaborating with cardiologists to develop a mobile app to help reduce hospital readmission rates for patients with congestive heart failure. This is accomplished through the creation of a cardiology knowledgebase and use of the explanation and discrimination inference tasks to recognize a person's health condition and suggest subsequent actions.

In the future, we plan to investigate more expressive approaches to explanation (beyond the single-feature assumption), rank explanatory features based on likelihood and/or severity, and rank discriminating properties based on their ability to reduce the number of explanatory features. In addition, we plan to extend our approach to stream reasoning by incorporating (i) periodic sampling and updating of observations, and (ii) explaining observations within a time window.

As the number and ubiquity of sensors and mobile devices continue to grow, the need for computational methods to analyze the avalanche of heterogeneous sensor data and derive situation awareness will grow. Efficient and scalable approaches to semantics-based machine perception, such as ours, will be indispensable.

Acknowledgements. This research was supported in part by US NSF award no. 1143717 (III: EAGER — Expressive Scalable Querying over Integrated Linked Open Data). We also thank Michael Cooney for help with implementation and evaluation.

References

1. Neisser, U.: Cognition and Reality. Psychology, p. 218. W.H. Freeman and Company, San Francisco (1976)
2. Gregory, R.L.: Knowledge in perception and illusion. Philosophical Transactions of the Royal Society of London 352(1358), 1121–1127 (1997)
3. W3C Semantic Sensor Network Incubator Group (SSN-XG) Charter, `http://www.w3.org/2005/Incubator/ssn/charter`
4. Lefort, L., et al.: Semantic Sensor Network XG Final Report. W3C Incubator Group Report (June 28, 2011), `http://www.w3.org/2005/Incubator/ssn/XGR-ssn-20110628`
5. Compton, M., et al.: The SSN Ontology of the W3C Semantic Sensor Network Incubator Group. Journal of Web Semantics (in press, 2012)
6. Gray, A.J.G., García-Castro, R., Kyzirakos, K., Karpathiotakis, M., Calbimonte, J.-P., Page, K., Sadler, J., Frazer, A., Galpin, I., Fernandes, A.A.A., Paton, N.W., Corcho, O., Koubarakis, M., De Roure, D., Martinez, K., Gómez-Pérez, A.: A Semantically Enabled Service Architecture for Mashups over Streaming and Stored 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. 300–314. Springer, Heidelberg (2011)
7. Calbimonte, J.P., Jeung, H., Corcho, O., Aberer, K.: Semantic Sensor Data Search in a Large-Scale Federated Sensor Network. In: 4th Intl. Workshop on Semantic Sensor Networks, Bonn, Germany, October 23-27 (2011)
8. Pfisterer, D., et al.: SPITFIRE: toward a semantic web of things. IEEE Communications Magazine 49(11), 40–48 (2011)
9. Sheth, A., Henson, C., Sahoo, S.: Semantic Sensor Web. IEEE Internet Computing 12(4), 78–83 (2008)

10. Ali, S., Kiefer, S.: μOR – A Micro OWL DL Reasoner for Ambient Intelligent Devices. In: Abdennadher, N., Petcu, D. (eds.) GPC 2009. LNCS, vol. 5529, pp. 305–316. Springer, Heidelberg (2009)
11. Shanahan, M.P.: Perception as Abduction: Turning Sensor Data into Meaningful Representation. Cognitive Science 29, 103–134 (2005)
12. Reggia, J.A., Peng, Y.: Modeling Diagnostic Reasoning: A Summary of Parsimonious Covering Theory. Computer Methods and Programs Biomedicine 25, 125–134 (1987)
13. Henson, C., Thirunarayan, K., Sheth, A., Hitzler, P.: Representation of Parsimonious Covering Theory in OWL-DL. In: 8th Intl. Workshop on OWL: Experiences and Directions, San Francisco, CA, USA, June 5-6 (2011)
14. Henson, C., Sheth, A., Thirunarayan, K.: Semantic Perception: Converting Sensory Observations to Abstractions. IEEE Internet Computing 16(2), 26–34 (2012)
15. Tobies, S.: Complexity Results and Practical Algorithms for Logics in Knowledge Representation. Ph.D. Thesis, RWTH Aachen, Germany (2001)
16. Bajcsy, R.: Active perception. IEEE 76(8), 996–1005 (1988)
17. Henson, C., Thirunarayan, K., Sheth, A.: An Ontological Approach to Focusing Attention and Enhancing Machine Perception on the Web. Applied Ontology 6(4), 345–376 (2011)
18. Kuhn, W.: A Functional Ontology of Observation and Measurement. In: Janowicz, K., Raubal, M., Levashkin, S. (eds.) GeoS 2009. LNCS, vol. 5892, pp. 26–43. Springer, Heidelberg (2009)
19. Devaraju, A., Kuhn, W.: A Process-Centric Ontological Approach for Integrating Geo-Sensor Data. In: 6th Intl. Conf. on Formal Ontology in Information Systems, Toronto, Canada, May 11-14 (2010)
20. Keßler, C., Raubal, M., Wosniok, C.: Semantic Rules for Context-Aware Geographical Information Retrieval. In: Barnaghi, P., Moessner, K., Presser, M., Meissner, S. (eds.) EuroSSC 2009. LNCS, vol. 5741, pp. 77–92. Springer, Heidelberg (2009)
21. Scheider, S., Probst, F., Janowicz, K.: Constructing Bodies and their Qualities from Observations. In: 6th Intl. Conf. on Formal Ontology in Information Systems, Toronto, Canada, May 11-14 (2010)
22. Devaraju, A., Kauppinen, T.: Sensors Tell More than They Sense: Modeling and Reasoning about Sensor Observations for Understanding Weather Events. International Journal of Sensors, Wireless Communications and Control, Special Issue on Semantic Sensor Networks 2(1) (2012) ISSN: 2210-3279
23. Taylor, K., Leidinger, L.: Ontology-Driven Complex Event Processing in Heterogeneous Sensor Networks. 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. 285–299. Springer, Heidelberg (2011)
24. Elsenbroich, C., Kutz, O., Sattler, U.: A case for abductive reasoning over ontologies. In: Workshop on OWL: Experiences and Directions, Athens, GA, USA, November 10-11 (2006)
25. Peraldi, S.E., Kaya, A., Möller, R.: Formalizing multimedia interpretation based on abduction over description logic aboxes. In: 22nd Intl. Workshop on Description Logics, Oxford, UK, July 27-30 (2009)
26. Binary RDF Representation for Publication and Exchange (HDT). W3C Member Submission (2011), `http://www.w3.org/Submission/2011/SUBM-HDT-20110330/`
27. Seitz, C., Schönfelder, R.: Rule-Based OWL Reasoning for Specific Embedded Devices. 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. 237–252. Springer, Heidelberg (2011)
28. Preuveneers, D., Berbers, Y.: Encoding Semantic Awareness in Resource-Constrained Devices. IEEE Intelligent Systems 23(2), 26–33 (2008)
29. Motik, B., Horrocks, I., Kim, S.: Delta-Reasoner: a Semantic Web Reasoner for an Intelligent Mobile Platform. In: 21st International World Wide Web Conference (WWW 2012), Lyon, France, April 16-20 (2012)
30. McGlothlin, J.P., Khan, L.: Materializing and Persisting Inferred and Uncertain Knowledge in RDF Datasets. In: 24th AAAI Conf. on Artificial Intelligence, Atlanta, GA, USA, July 11-15 (2010)

Semantic Enrichment by Non-experts: Usability of Manual Annotation Tools

Annika Hinze[1], Ralf Heese[2], Markus Luczak-Rösch[2], and Adrian Paschke[2]

[1] University of Waikato
hinze@waikato.ac.nz
[2] Freie Universität Berlin
{heese,luczak,paschke}@inf.fu-berlin.de

Abstract. Most of the semantic content available has been generated automatically by using annotation services for existing content. Automatic annotation is not of sufficient quality to enable focused search and retrieval: either too many or too few terms are semantically annotated. User-defined semantic enrichment allows for a more targeted approach. We developed a tool for semantic annotation of digital documents and conducted an end-user study to evaluate its acceptance by and usability for non-expert users. This paper presents the results of this user study and discusses the lessons learned about both the semantic enrichment process and our methodology of exposing non-experts to semantic enrichment.

1 Introduction

Current internet search engines typically match words syntactically; semantic analysis is not supported. The Semantic Web envisions a network of semantically-enriched content containing links to explicit, formal semantics. So far, most of the semantic content available has been generated automatically by either wrapping existing data silos or by using annotation services for existing content. We believe, however, that the success of the Semantic Web depends on reaching a critical mass of users creating and consuming semantic content. This would require tools that hide the complexity of semantic technologies and match the compelling simplicity of Web 2.0 applications: light-weight, easy-to-use, and easy-to-understand. Very little research has been done on supporting non-expert end-users in the creation of semantically-enriched content.

We studied the process of manual creation of semantic enrichments by non-experts. For this, non-expert users were observed interacting with an example annotation system. We used *loomp*, an authoring system for semantic web content [1]. In its initial design phase, *loomp* received positive feedback from non-expert users (e.g., journalists and publishing houses). Their feedback revealed great interest in adding "metadata" to content but also some difficulty in understanding the underlying principles of semantic annotations. This motivates research to derive guidelines for the design of adequate annotator tools for non-experts and to gain insight into non-experts' understanding of semantic annotations. To explore the experience of non-experts in operating a semantic annotation tool, we therefore conducted a user study of the *loomp* annotator component. This paper reports on the results of this user study and discusses its implications for the semantic web community. We will argue that manual semantic annotations need specialized task experts (instead of domain experts) and we note a lack of clearly defined use cases and accepted user-centred quality measures for semantic applications.

The remainder of this paper is structured as follows: In Section 2 we explore related work in evaluating the usability of semantic web tools. Section 3 introduces *loomp* and the *One Click Annotator*. The methodology and setup of our study is explained in Section 4. Section 5 reports on the results of the study, while Section 6 discusses the implications of the study results. Section 7 summarizes the contributions of our study and draws conclusions for semantic web tool design.

2 Related Work

Here we discuss research related to the *loomp OCA* and its evaluation.

Annotation Tools. Annotation can be done either automatically or manually (or in combination). Automatic annotation tools are typically evaluated only for precision and recall of the resulting annotations [2,3,4]. Most manual annotation tools have never been evaluated for their usability; many are no longer under active development [5,6,7]. We classify current manual systems into commenting tools [8,9,10], web-annotation tools [11,12], wiki-based systems [13,14,15], and content composition systems [1,16], digital library tools [17,18,19], and linguistic text analysis [20].

Naturally, due to their different application fields, the tools encounter different challenges in interaction design and usability (e.g. wiki tools require users to master an annotation-specific syntax and to cope with many technical terms). However, we believe the most significant challenge for user interaction design is defined by the conceptual level of semantic annotation. That is, the annotation process is conceptually different if tools support simple free-text annotation (e.g.,[8,9,10]), offer a shared vocabulary of concepts (e.g., [6,7,11]), use a local semantic identity (by thesaurus or ontology, e.g., [5,13,15]), or use shared semantic identity (e.g., by linked ontologies referencing with a linked data server such as DBpedia, e.g., [2,3,18]). The development of most annotation tools has a strong focus on providing novel functionality. For the manual annotation tools, usability was typically a factor in the interface development. However, end-user evaluations of interface and user interaction are very rare.

End-User Experience of Annotations. Few system evaluations have considered end-user experiences. Handschuh carried out a tool evaluation with user involvement; however, participants were used merely to provide annotation data that was then analysed for inter-annotator correlation [7,21]. Furthermore, the researchers expressed disappointment about the low quality of annotations. Feedback on the participants' experience and their mental model of the process were not sought. Bayerl et al. [22] stresses the importance of creating schemas and systems that are manageable for human annotators. They developed a method for systematic schema development and evaluation of manual annotations that involves the repeated annotation of data by a group of coders. Erdmann et al. [23] performed studies on manual and semi-automatic annotation involving users. They describe their participants as "more or less" able to annotate webpages. However, the majority of issues identified were of a syntactic nature that could easily be remedied by tool support. Work on rhetorical-level linguistic analysis of scientific texts is closely related to semantic annotation [20]. Teufel performed user studies in which she looked for stability (same classification over time by same annotator) and reproducibility (same classification by different annotators; similar to Handschuh's inter-annotator correlation). Similar to [22], she found that complex schemas may lead to lower quality annotations, and subsequently simplified the predefined set of common concepts that was used in the evaluation (from 23 to 7 concepts). Teufel assumed that annotators would be task-trained and familiar with the domain. We discuss observations of these studies relating to semantic understanding in Section 6.

Benchmarking. A number of researchers have discussed methodologies for comparing annotation tools using benchmarks [24,25]. Maynard developed a set of evaluation criteria for performance as well as for usability, accessibility and inter-operability [24,26]. However, usability here refers to practical aspects such as ease of installation and online help, and does not contain concepts of interaction design, user acceptance and effectiveness. Schraefel and Karger [27] identify ontology-based annotation as one of the key concepts of SW technologies and defined a set of quality criteria. One of these is usability, which for them covers ease-to-learn, ease-of-use and efficiency. Uren et al. [28] developed a survey of annotation tools in which "user-centered collaborative design" was one of the requirements. However, they mainly explore the ease-of-use of tool integration into existing workflows. They furthermore assumed that annotation would be created by "knowledge workers." Most benchmarks focus on (user-independent) performance measures; usability concepts are seldom included and rarely evaluated. Castro [25] observes that in the semantic web area, technology evaluation is seldom carried out even though a number of evaluation and benchmark frameworks exist.

HCI Challenges in the Semantic Web. A series of workshops on Semantic Web HCI identified areas for research contribution, one of which is the capture of semantically-rich metadata without burdening the user [26]. Karger suggests hiding the complexity of the Semantic Web by developing tools that look like existing applications and to develop better interfaces to bring the semantic web forward "before AI is ready" [29]. Jameson addresses a number of concerns of the SW community about user involvement and stresses the value of both positive and negative results of user studies [30].

Ontology Engineering. The development of ontologies faces similar challenges to that of the semantic annotation of texts: It is a complex task that often needs (manual) user input [31,32,33]. However, ontology engineering is typically executed by experts in semantic technologies and is not necessarily suitable for end-users. However, Duineveld at al. [31] report that often the bottleneck in building ontologies still lies in the social process rather than in the technology. User-oriented evaluation focuses predominantly on syntactic problems (e.g., how much knowledge of the representation language is required), but not on conceptual questions such as the user's mental models of the system.

Summary. Even though aspects of HCI and user involvement have been identified as important aspects for Semantic Web technologies, typical benchmarks and evaluation strategies do not contain complex user aspects. Few studies involving end-users have been executed in the context of semantic annotations. In particular, manual annotation tools have so far not been systematically evaluated for appropriate interaction design and semantic understanding. System evaluations that incorporated human participants did not seek their feedback on interaction issues nor did they evaluate the participants' mental models of the system interaction. So far, issues of understanding of semantic annotations by (non-expert) users have not been studied in a systematic manner.

3 Semantic Content Enrichment Using *loomp*

loomp is a light-weight authoring platform for creating, managing, and accessing semantically enriched content. Similarly to content management systems allowing people unfamiliar with HTML to manage the content of websites, *loomp* allows people unfamiliar with semantic technologies to manage semantically enriched content. *loomp*'s *One Click Annotator* enables users to add semantic annotations to texts. We first discuss user-related design considerations for the *loomp OCA*, and then briefly give an introduction into the concepts of the *OCA*.

3.1 User-Oriented Design Considerations

A key goal for *loomp OCA* was to hide the complexity of semantic technologies (cf. [29]) but nevertheless allow for the creation of meaningful and consistent annotations. We identified following key requirements for non-expert users.

Established interaction patterns. Karger argues that there is an advantage of making Semantic Web applications look like existing applications and to use familiar interaction paradigms. [29,27]. A similar argument has been made in the area of personal information management, where new tools are more successful if they are able to extend the functionality of existing applications rather than introducing an entirely new way of doing things [34]. For the *loomp One Click Annotator*, we therefore adopt well-known interaction procedures of widely used software (such as text highlighting and formatting in MS Word^{TM}).

Simple vocabularies. It has been shown that complex thesauri and category structures are disadvantageous for quality annotations [22,20]. For a given task, users may only require a small part of a vocabulary that is modeled in a large ontology with deep hierarchical structures. Thus, *loomp OCA* only offers an appropriate subset of terms and provides support in choosing the right annotation.

Contextual semantic identity. The RDF data model differs in subtle ways from the cognitive models that humans create for the content of a text. In RDF, resources are assigned URIs for unique identification and disambiguation of concepts and instances. Although people may recognize URLs as addresses of websites, they are not used to identifying real-world entities by URL/URIs and are typically not familiar with the concept of namespaces. Instead, humans use labels to refer to entities (e.g., "baker") and disambiguate meaning by the textual context (e.g., as reference to the person Baker or the profession baker). The annotator has to bridge this gap between objective knowledge (as encoded in the RDF data model) and subjective knowledge of human cognition [35,36]. The *loomp OCA* aims to support this process by presenting labels and contextual information for identification of semantic identity.

Focus on the user's task. Handschuh and Staab observed that semantic authoring and semantic annotations have to go hand in hand [37]. As a consequence, we integrated the

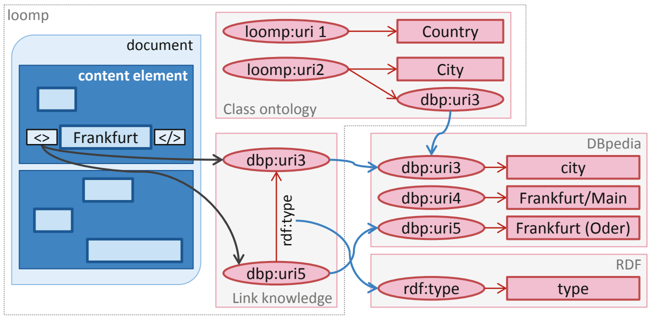


Fig. 1. *loomp OCA*: Conceptual data model

loomp OCA toolbar for creating semantic annotations seamlessly into the *loomp* editor, so that the user can add annotations without being distracted from their primary task.

3.2 *loomp OCA* Conceptual Design

loomp's domain model consists of content elements, ontological concepts and annotations that create links between them, as well as encapsulating documents. Each document consists of a sequence of content elements (see Fig. 1), where a content element can belong to several documents. We use DBpedia as the base ontology to identify annotation concepts (classes) and instances.

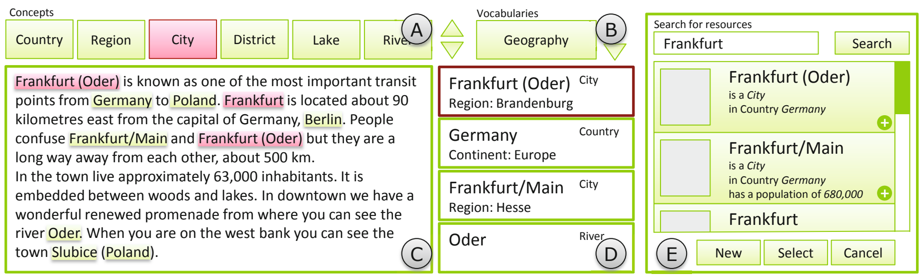
The *loomp OCA* offers a selection of class ontologies (subsets of DBpedia) to present vocabularies to the users (see Fig. 2, B and C).

A user can assign an annotation to a text atom (i.e. a part of a content element) in two steps.

1. The user marks an atom in the text field and then selects a concept from an offered vocabulary (see Fig. 2, B). For example, they mark the atom Frankfurt and select the concept City from the vocabulary Geography. Internally, the system then creates a link between the atom and the concept id, which is inserted into the content element as RDFa element (transparent to the user).
2. The system sends the text of the atom as a query to DBpedia. The labels of the resulting instances are filtered by the concept, and then presented to the user for selection. For example, the system sends Frankfurt as a query to DBpedia, where all instances containing this phrase are identified. The result set is then filtered by the concept term city. The user is presented with the resulting instance list containing references to Frankfurt/Oder and Frankfurt/Main (see Fig. 2, right). They identify the appropriate instance to be linked to the atom. Internally, the system creates a second link for the atom, linking to the instance id (here, linking to the DBpedia id of Frankfurt (Oder), see Fig. 1).

The creation of the links from the atom to both concept id and instance id allows identification of link knowledge, such as the type of the instance resource (Frankfurt (Oder) rdf:type City[1]). As a result, when linking atoms from different documents to the same

[1] For example, encoded as (dbp:uri5 rdf:type dbp:uri3), using ids from Fig. 1.



Frankfurt (Oder) is known as one of the most important transit points from Germany to Poland. Frankfurt is located about 90 kilometres east from the capital of Germany, Berlin. People confuse Frankfurt/Main and Frankfurt (Oder) but they are a long way away from each other, about 500 km.
In the town live approximately 63,000 inhabitants. It is embedded between woods and lakes. In downtown we have a wonderful renewed promenade from where you can see the river Oder. When you are on the west bank you can see the town Slubice (Poland).

Frankfurt (Oder) City
Region: Brandenburg

Germany Country
Continent: Europe

Frankfurt/Main City
Region: Hesse

Oder River

Frankfurt

Frankfurt (Oder)
is a *City*
in Country *Germany*

Frankfurt/Main
is a *City*
in Country *Germany*
has a population of *680,000*

Frankfurt

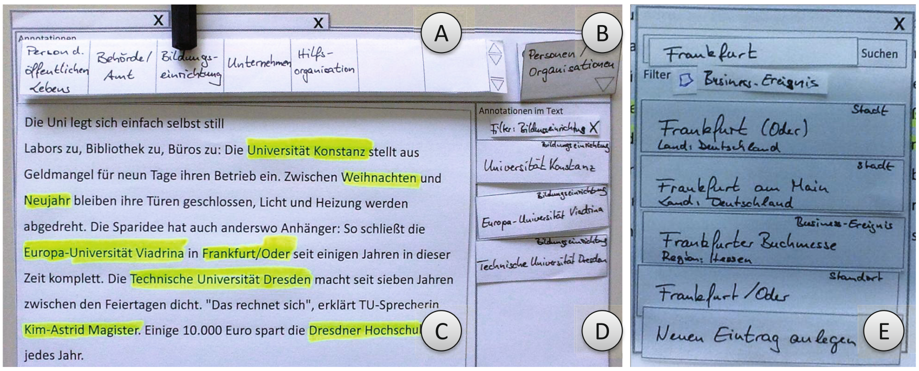
Fig. 2. Interface of *loomp OCA*

semantic identifier (e.g., to the DBpedia id of Frankfurt (Oder)), a user creates conceptual cross-links between these documents. As this paper focuses on the user interaction with the *loomp OCA*, we refer for technical details of *loomp OCA* to [1,38].

4 User Study Methodology

We studied the usability of the *loomp OCA* in an end-user study. Following Degler [39], who evaluated methods for improving interaction design for the Semantic Web, we performed usability tests and interviews with real users (not student stand-ins). The aim of our study was to (1) evaluate the suitability of the tool for non-experts, and (2) explore how non-expert users experience and apply the concept of annotating texts. Even though the *loomp* system is fully operational, the user study was executed with a paper prototype. This allowed us to gather feedback from non-expert users in a non-threatening technology-free context. A paper prototype consists of mock-up paper-based versions of windows, menus and dialog boxes of a system. One of two researchers plays the role of the 'system', the other one acts as facilitator. Participants are given realistic tasks to perform by interacting directly with the prototype – they "click" by touching the prototype buttons or links and "type" by writing their data in the prototype's edit fields. The facilitator conducts the session; the participants are video-taped and notes are taken. The 'system' does not explain how the interface is supposed to work, but merely simulates what the interface would do. In this manner, one can identify which parts of the interface are self-explanatory and which parts are confusing. Because the prototype is all on paper, it can be modified very easily to fix the problems. Paper prototyping is an established usability method, which has been shown to allow greater flexibility in reacting to user activities and to elicit high quality and creative feedback as users do not feel restricted by an apparently finished software product [40]. The user study was set up as follows:

Paper prototype. Mirroring the *loomp OCA* interface, the paper prototype consisted of two windows (see Fig. 3). All UI components of the functional software described in Section 3 are present: (A) the text pane, (B) the annotation toolbar consisting of two parts, (C) the annotation sidebar, and (D) the resource user (as separate pop-up window). The framework of the user interface and outlines of interaction elements were printed on paper and cut to size; alternatives and pull-down menus were simulated by folding the paper into concertinas. All labels on interaction elements were hand-written to allow



Die Uni legt sich einfach selbst still

Labors zu, Bibliothek zu, Büros zu: Die Universität Konstanz stellt aus Geldmangel für neun Tage ihren Betrieb ein. Zwischen Weihnachten und Neujahr bleiben ihre Türen geschlossen, Licht und Heizung werden abgedreht. Die Sparidee hat auch anderswo Anhänger: So schließt die Europa-Universität Viadrina in Frankfurt/Oder seit einigen Jahren in dieser Zeit komplett. Die Technische Universität Dresden macht seit sieben Jahren zwischen den Feiertagen dicht. "Das rechnet sich", erklärt TU-Sprecherin Kim-Astrid Magister. Einige 10.000 Euro spart die Dresdner Hochschul jedes Jahr.

Fig. 3. Paper prototype of the *loomp OCA*

dynamic changes. The available texts to annotate were printed onto text pane templates. Designing the paper prototype in this way allowed us to react easily to unexpected user behaviour (e.g., by creating resources for unexpected annotations) and to make small changes to the user interface on the fly.

The participants received a marker pen to simulate the use of a computer mouse (used for highlighting text in the text pane and selecting UI elements by clicking with closed pen). This simulated mouse was readily accepted by the users; some additionally invented right clicks and Alternate keys. The fast changing highlighting of UI elements (indicated by a pressed button and colour change in the *loomp* software) were indicated by pen caps being placed onto the elements (see top left of Fig. 3).

Texts and Ontology. We prepared two texts for annotation that contained only general knowledge concepts. Thus every participant was a domain expert. The first document was used as a training set; it contained a short text about Konrad Adenauer, the first chancellor of West Germany. This allowed the participants to explore the interface without any pressure to "get it right." The second, longer text was about the weather and universities being closed during the cold period. Both texts were based on news portal entities that were shortened for the study. We adapted the texts so as to explore interesting semantic problems, such as place names with similar words (Frankfurt (Oder) and Frankfurt/Main), nested concepts (Universität Konstanz) and fragmented references (Konrad Hermann Joseph Adenauer and Konrad Adenauer referring to the same person). These adaptations ensured that participants had to select the correct resources to link to an atom. We used the same underlying 'news' ontology for both texts. A subset of classes of this ontology was selected manually to provide a set of concepts tailored to the example texts (while allowing for variations). The classes were grouped into three named vocabularies: Persons & Organizations, Events, and Geography. They contained 12, 8, and 10 annotations respectively. Identical underlying ontology and annotations sets were used for learning and application phase.

Study phases. The study was performed by two researchers: the first interacted with the participants, while the second acted as system simulator (no direct interaction between participants and second researcher). The study was performed in four phases: introduction, learning phase, application phase, and guided interview. During the introduction,

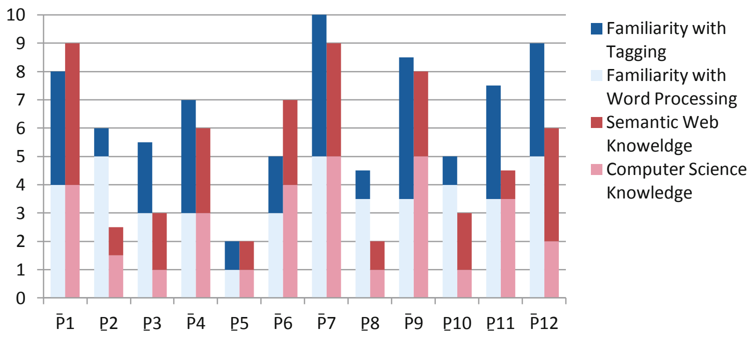


Fig. 4. Participants' self assessment

the aim of the project and the prototype were explained and the participant was shown the paper prototype. During learning phase and application phase, the participant interacted with the prototype. The researchers took notes and recorded the interactions. In the learning phase, the researcher explained the purpose of the application by way of a use case for semantic search and thus illustrated the need for semantic annotations. The participant received instructions on the task of annotating. Participants were given time to familiarize themselves with both the task and the interface. In the application phase, the longer text was used with the same task. The participants were encouraged to think out loud while making decisions in interaction with the prototype, instead of asking for the 'correct' procedure. The study had 12 participants (up to 1.5 hours interaction each).

5 Results

We here describe our observations of how participants interacted with the *One Click Annotator*. We differentiate between general observations about the participants (Sect. 5.1), observations related to the interaction with UI elements (Sect. 5.3) and observations related to the task of creating semantic annotations (Section 5.4). Implications for tool design and the Semantic Web community will be discussed in Section 6.

5.1 Participant Demographics

As the tool is designed for non-experts, we selected 12 participants with varied backgrounds (e.g., librarians, PR managers, secretaries). We enquired about their familiarity with word processing (as a measure of computer literacy), tagging (as an annotation task), computer science and Semantic Web (as technical expertise). Participants rated their knowledge on a 5-point scale (1=no knowledge, 5=very knowledgeable). Fig. 4 shows the distribution of expertise for the 12 participants. 11 of 12 participants are computer literate (basic computer literacy is a requirement for *loomp*), six are familiar with tagging and setting annotations and thus have advanced computer literacy skills. Six participants have very little knowledge in computer science and Semantic Web; they are (technical) non-experts – the user group for which *loomp* was designed. Based on their self-assessment, we identified participants P̄1, P̄4, P̄6, P̄7, P̄9 and P̄12 as *technical* *experts* (CS+SW knowledge ≥ 6) and participants P̲2, P̲3, P̲5, P̲8, P̲10 and P̲11 as *non-experts* (CS+SW knowledge < 5). Throughout the paper, we visually indicate expertise thus: P̲x and P̄x. We observe that technical experts are also (highly) computer literate.

5.2 UI Elements: Observed Interaction Patterns

We now describe participant interactions with the key elements of the *loomp OCA* with a focus on the participants' understanding of the annotation process.

i) Annotation toolbar (A+B in Fig.3). Some participants had difficulties interacting with the annotation toolbar. Some participants selected first an annotation without highlighting atoms. P̄1 had forgotten to select an atom first; P̄12 intended to use the annotation as a paint brush to assign it to several atoms. A number of participants had difficulty remembering the available concepts they had just looked at.

ii) Text pane (C). Most participants had no problems selecting atoms to assign annotations. Five of 12 participants tried to create nested annotations, which is currently not supported in *OCA*. Taking the atom Universität Konstanz as an example, they wanted to assign Educational institution to the phrase and City to the term Konstanz. Two participants (P̄4, P̲10) lost the annotation of the city because the later assignment of the larger atom Universität Konstanz overwrote the previous smaller atom Konstanz. Only P̄4 understood and corrected the mistake. P̲10 observed the loss of annotation but did not recognize the problem. In contrast, two participants allocated Educational institution before allocating City with the result that the first annotation covered only the atom Universität. Five of 12 participants wanted to assign more than one annotation to the same atom, e.g., Adenauer is a person and a politician. Participant P̄12 wanted to use the ALT-key to select all text occurrences of an atom and then select the annotation only once to link all occurrences to the same resource (e.g., all occurrences of the same city name). The same participant suggested that the system should also support the following process: Users highlight all text occurrences of entities of the same type (e.g., all cities), click on an annotation button, and choose a resource to each of these one after the other.

iii) Annotation sidebar (D). Participants used the annotation sidebar either as a simple list for compiling the created annotations, or as a tool for highlighting occurrences of created annotations (P̄1, P̄7) and for copying existing annotations to newly selected atoms (P̄9, P̲11). P̄9 also clicked on the concept (e.g., Person) of an annotation shown in the sidebar and, thus, filtered the entries of the sidebar according to that concept. Participants did not recognize the temporal order of annotations in the sidebar (Participant P̄4 suggested they be sorted alphabetically).

iv) Resource selector (E). Several participants had difficulties in understanding the resource selector. The selector recommends resources for a selected atom, which was readily accepted by the participants without reflecting on the source of the recommendations. The recommendations are created based on DBpedia search results for the atom text. As a consequence, generic atom names (e.g., university) lead to long result lists (P̄7 was so overwhelmed by the long result list that he closed the window without selecting an entry). Only five of the participants (P̄1, P̄6, P̄7, P̄9, P̲10) recognized the text field as a means for searching for resources manually and only two of them (P̄7, P̲10) understood the filter option below the search field.

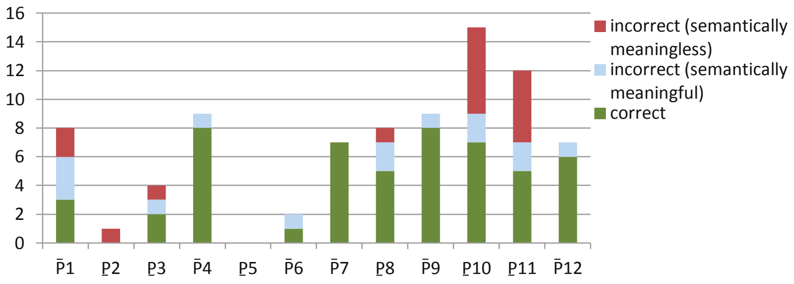


Fig. 5. Correctness of Annotations

5.3 Task: Understanding Semantic Concepts

We were interested in participants' conceptual understanding of the creation of semantic annotations. We evaluated the selected atoms and annotations, as well as the participants' reaction to the resources recommended in the resource selector.

i) Quality of Annotations. Fig. 5 shows an analysis of the annotations created by the participants (second phase only). The gold-standard annotation for the text contained eight annotations (for the given vocabulary). An annotated atom is considered semantically meaningful if it refers to a named entity, e.g., if participants allocated City to the atom Dresden within Technische Universität Dresden. Finally, annotations are considered semantically meaningless if they do not refer to a named entity, e.g., light and heating are turned off. Six of 12 participants identified at least six correct annotations and another one created 5 of 8 correct annotations. However, two participants additionally created many semantically meaningless annotations (P̲10,P̄12). P̲2 and P̲5 failed to create any meaningful annotations.

ii) Assuming system knowledge. Six participants switched from being an information provider to an information consumer in the course of the study (P̲2, P̲3, P̲5, P̲8, P̄9, P̲11). (P̲3 "Now I want to know something about his political career;" P̲2: "Now I cannot find any information"). Three of them clicked concepts without selecting any text because they expected that the system would highlight the respective named entities, e.g., highlight all mentioned cities. Five participants assumed that the system comes with background knowledge (e.g., P̲8 clicked on the term 'chancellor' and said "There should be a list of chancellors of FRG.")

iii) Selecting annotations. In the first annotation step (see Section 3.2), four participants wanted to create a summary of the text by selecting whole sentences as atoms (P̄1, P̲2, P̲3, P̄6). For example, P̄1 selected light and heating are turned off and allocated the annotation Political Event. P̄6 comented that "Somehow the whole text is an event."The selection of unsuitable atoms resulted in difficulties when selecting the correct annotation: P̲10 selected the atom library and allocated Educational Institution. She observed: "I find it difficult to find the correct annotation." She proceeded similarly with laboratory and office. Several participants aimed to annotate such classes of instances (in addition to instances), which almost always led to the unnecessary creation of new resources. We also observed difficulties when one concept is a subclass of another one (e.g., Person and Politician). As the prototype did not support overlapping annotations, almost all participants chose the more specific concept. Only P̄4 explained that he assumed the system would contain information about the relationship between the two concepts. In contrast, three participants (P̄7, P̄9, P̲10) developed a workaround and annotated one occurrence with Person and another one with Politician. Four participants had difficulty deciding on the granularity of atoms (P̄4, P̲8 P̄9, P̲10), e.g., whether to annotate the city in the atom Technische Universität Dresden.

iv) Interaction patterns. We observed different strategies for creating annotations. Two participants (P̲8, P̄12) first seemed to select a vocabulary and then scan the text for occurrences of matching atoms, e.g., to annotate all cities (P̄12: "The cities are done now."). P̄12 suggested having an "annotation painter" (similar to the format painter in office software) that allows for selecting a concept from a vocabulary and then selecting atoms. Another common strategy was to annotate an entity and search for further occurrences in the text. A few participants felt unsure whether they had created enough annotations, e.g., P̄7 commented "I would ask a colleague."

v) Identifying entities. In the second annotation step (see Section 3.2) of linking atom to resource, we observed problems in choosing the appropriate entry in the resource selector. P̄9 wanted to annotate the name of the river Oder in the name of the city Frankfurt/Oder. The resource selector offered two rivers, one of them within the correct region. P̄9 wondered: "Why is that, is that one river or two?" and continued by creating a new resource. However, all five participants annotating Frankfurt/Oder successfully selected Frankfurt (Oder) as resource (i.e., it was clear that both labels referred to the same city).

vi) Additional knowledge work. During the learning phase, five participants wanted to insert additional information, e.g., create cross references or even extend a vocabulary. For example, P̄1 wanted to relate Kim Astrid Magister with Technical University Dresden because she was the spokeswoman of that university. Later, while annotating the term Christmas the same participant stated: "I want to create synonyms because I can create a larger vocabulary faster." Another participant wanted to convert units, e.g., 150 kmph to mph. P̄6 was not satisfied with the available vocabularies and wanted to add his own.

5.4 Reflection: Participant Feedback

We interviewed the participants about their experience in using the *loomp OCA*. Fig. 6 shows the participants' self-assessment regarding their mastery of annotations (1=no knowledge, 5=completely understood). These results are also interesting in the light of the quality of the created annotations (see Fig. 5).

We asked the participants for feedback on their *understanding* of annotations (left), ease of *creating* annotations (middle), and the ease of creating the *right* annotation (right). 9 participants found annotations easy to understand, 7 found it easy to create annotations, and 3 found it easy to create the right annotations. On average, expert participants (P̄1, P̄4, P̄6, P̄7, P̄9, P̄12) found annotations easier to understand (4.42 vs 4.0) and create (4.17 vs 3.33) than non-experts. However, both experts and non-experts found it somewhat difficult to select the right annotations (both 3.33).

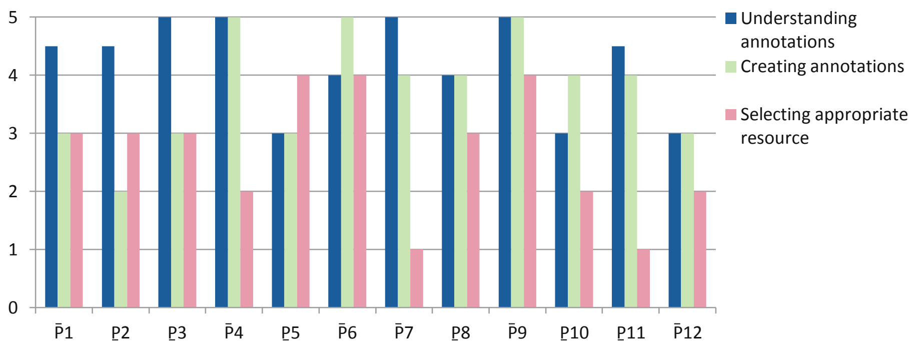


Fig. 6. Self-assessment on understanding and using annotation concepts

6 Discussion

We now discuss the insights gained from the study and draw conclusions for the design of manual annotation tools for the semantic web. We distinguish between Semantic Web 2.0 approach to annotations and the closed world of a corporate context. People responsible for creation of corporate content are typically domain experts but non-experts with respect to semantic web.

Task Understanding. Some participants expected the system to be all-knowing and all-powerful. This well-known aspect from novice computer users (cf. [41]) here applied to non-experts with respect to semantic technologies. They assumed background knowledge about people mentioned in the documents, as well as complex semantic relationships between concepts in the vocabulary. This was tightly interwoven with the problem of participants' switching roles from information provider to information consumer. The task of providing conceptual, semantic information seemed so foreign to some of the participants that eventually ingrained habits of information consumers came to the fore; see Sect. 5.2/ii. This was different for participants with a strong CS/SW background; they created higher quality annotations (5.2/i) and also felt more confident about the task (5.4). However, these individuals would not be suitable as knowledge workers in a corporate environment (too highly skilled technically), but could be for an open crowd-sourcing approach.

Suitability of Non-experts. Based on annotation results, we identified three groups of participants: acceptable annotators ($\bar{P}$4, $\bar{P}$7, $\bar{P}$9, $\bar{P}$12), annotators with room for improvement ($\bar{P}$1, $\underline{P}$8, $\underline{P}$10, $\underline{P}$11), and failed annotators ($\underline{P}$2, $\underline{P}$3, $\underline{P}$5, $\bar{P}$6), see Sect. 5.3/i-iii. We hypothesize that the results for the first two groups may be improved by further instructions on correct annotation. We do not believe that participants in the last group are suited as annotators. This is surprising as some were in professions that required regular creation of keywords and tagging for books (e.g., $\underline{P}$5: librarian). We note that the participants in the group of acceptable annotators were all technical experts. Comparing their individual backgrounds we observed that participants of the second had more experience in working with text, in managing knowledge, and were more open to new conceptual ideas those in the third group. Furthermore, technical knowledge does not always guarantee high quality annotations (see Sect. 5.3/i). We conclude that semantic annotations cannot be done by *domain experts* (as typically assumed in the SW context) but needs *task experts*. These would be familiar not just with the domain (as all our participants were) but also with the subtleties of the annotation task.

User Guidance through Vocabularies. Even though the three provided vocabularies were relatively small, it was difficult for some participants to know which concepts to select and when to stop (5.2/i+5.3/iv). We concur with observations in [22,20], and observe that a reduction in size of available vocabularies and annotations helps to keep annotators focused on identifying (named) entities and will increase the quality of annotations. However, reducing the size of the vocabulary is not always a viable option and therefore documentation (SW2.0) and education (corporate) have to be explored. Dynamic online history in the annotation sidebar had mixed results (5.2/iii) and needs to be explored further.

Semantic Identity. All participants were able to select the correct resource from the list if the entries contained enough information to identify the entity they had in mind. Problems arose when participants were unable to disambiguate recommended entities. Only four technical experts and one non-expert recognized the search field and the filter in the Resource selector (5.2/iv). No correlation was detected between the understanding of the resource selector and the correctness of annotations (5.3/i+v). However, selection of atoms dominated the annotation process. If in doubt about atoms, participants created new resource ids. It highlights the importance of educating annotators with the conceptual model of semantic identity and its difference to tagging. We believe non-expert users need targeted teaching material with appropriate use cases highlighting the benefits of annotation-based applications.

Interaction Patterns Aligning to Mental Model. The paper prototype resembled the look and feel of MS Word (following [29]) to allow easy recognition of known interaction patterns (5.2/i+ii). However, we found that the participants' mental model of how the system works had strong references to Internet-based interactions, even though the interface did not contain any typical Web elements (5.3/ii+vi). P11 mentioned wikipedia search as a related concept. One reason for this mismatch may be the participants' (conceptual or practical) familiarity with web-based tagging. Thus it needs to be explored which references and expectations non-expert users bring to semantic web systems.

Corporate vs. Public Semantic Web. The usage context of annotators in a corporate or public setting may differ significantly (e.g., editing new text vs. annotation of existing text). Clear *use cases* may help explore these user expectations and usage contexts. Furthermore, questions of annotation quality are expected to be answered differently within each of these contexts (cf. 5.2/i). Corporate semantic web annotations would be expected to follow a pre-defined semantics with quality measures of inter-annotator correlation [7] and stability & reproducibility [20]. However, in the public Semantic Web 2.0 sector, variation in annotation may not only be permissible (5.2/iii) but sought as an opportunity to reflect multiple perspectives on a source (e.g., supporting the needs of vision-impaired people [4]).

Annotation Nesting. A number of participants wanted to create nested annotations (5.2/ii+5.3/iii). *loomp OCA* does not currently support nested annotations as it uses XHMTL+RDFa to include annotations into the texts. Overlapping annotations cannot easily be represented in XHTML as they they would result in ill-formed XML. Visualization of nested annotations is also challenging [42]. We therefore focused on the annotation process in a paper prototype according to *loomp OCA* and did not allow for nested annotations. However, the observations of our study clearly indicate a need for the support of nested annotations.

7 Conclusions and Future Work

In this paper, we described a user study to evaluate the conceptual understanding of semantic annotations by non-expert users. All participants of the user study were computer-literate domain experts, six of the 12 were non-experts with respect to semantic technologies.

The results of our user study indicated that not every domain expert is a good annotator due to difficulties in the understanding of the conceptual model of semantic annotations. Though some participants had familiarity with providing content and metadata (e.g., from their occupational background), many fell back into the role of content consumers and expected the editor to provide information. Because very few use cases and applications for non-experts require the creation of semantic annotations, we assume these participants were insufficiently accustomed to this task. Even though most participants readily understood the process of creating annotations, we observed a number of challenges: granularity of atoms (e.g., sentence, phrase, word), well-formed atoms (i.e referring to named entities), annotating both instances and concepts, complexity of vocabulary, and the tendency to create new resources even though an appropriate resource exists. Some participants wanted to create a summary or synonyms instead of annotations; they felt unsure if an annotation was useful or whether they had finished. We see the reasons for these difficulties predominantly in the lack of conceptual understanding, a lack of easy-to-understand use cases and in deficits in the interaction design.

Although the study used the graphical interface of the *loomp One Click Annotator*, our results can be transferred to other editors for manually or semi-automatically annotating contents by non-experts:

Task experts: Current literature distinguishes between technical experts and domain experts. Based on our study observations, we introduce the new concept of *task experts*. Task experts are domain experts who conceptually understand the task of annotating texts and have insight into the characteristics of semantic annotations (e.g., semantic identity).

Need for use cases: We note a lack of use cases illustrating the process of annotating texts and demonstrating the benefits of semantic annotations. Use cases may need to be customized to corporate or public semantic web context.

Semantic identity: For high quality annotations, users need help in selecting appropriate resources for linking. The recommendation algorithm therefore plays an important role, and needs to be supported by an appropriate interface representation of recommended resources to users. In particular, these need to take into account that users have difficulties distinguishing between instances and classes of instances.

User evaluation methodology: We noted a lack of commonly accepted quality measures for manual semantic annotation. Furthermore, there is a lack of clearly defined methodologies for evaluating the user aspects of semantic web applications.

We currently investigate user interaction with a variety of software prototypes for semantic annotation [42] as well as their implications for digital document repositories [43]. For future research, we plan to conduct further user studies on semantic annotation in more specific usage contexts, such as combined editing and annotating in a corporate setting. Furthermore, *loomp* users may first invoke an automatic annotation service and then revise the generated annotations manually (e.g., to resolve overlapping or ambiguous annotations). A similar approach of a-posteriori annotation is supported by RDFaCE [11]. We plan to evaluate the influence of such pre-existing annotation sets on the subsequent manual annotation.

We plan to additionally investigate alternative user interfaces for selecting annotations and resources. We are interested in the impact of clearly defined use cases with practical relevance and accompanying teaching material on the quality of annotations defined by non-experts.

References

1. Luczak-Rösch, M., Heese, R.: Linked data authoring for non-experts. In: Linked Data on the Web Workshop, World Wide Web Conference (April 2009)
2. Thomson Reuters Inc.: Open Calais website, `http://www.opencalais.com/`
3. Zemanta Ltd.: Zemanta (2012), `http://www.zemanta.com/`
4. Yesilada, Y., Bechhofer, S., Horan, B.: Cohse: Dynamic linking of web resources. Technical Report TR-2007-167, Sun Microsystems (August 2007)
5. Kalyanpur, A., Hendler, J., Parsia, B., Golbeck, J.: SMORE - semantic markup, ontology, and RDF editor. Technical Report ADA447989, Maryland University (2006)
6. McDowell, L., Etzioni, O., Gribble, S.D., Halevy, A., Levy, H., Pentney, W., Verma, D., Vlasseva, S.: Mangrove: Enticing Ordinary People onto the Semantic Web via Instant Gratification. In: Fensel, D., Sycara, K., Mylopoulos, J. (eds.) ISWC 2003. LNCS, vol. 2870, pp. 754–770. Springer, Heidelberg (2003)
7. Handschuh, S., Staab, S.: Cream: Creating metadata for the semantic web. Computer Networks 42(5), 579–598 (2003)
8. Booktate: Booktate website (2012), `http://booktate.com/`
9. Textensor Limited: A.nnotate (2012), `http://a.nnotate.com`
10. Olive Tree Bible Software, Inc.: Bible+ (2012), available online at `http://itunes.apple.com/`
11. Khalili, A., Auer, S.: The RDFa content editor - from WYSIWYG to WYSIWYM (2011), `http://svn.aksw.org/papers/2011/ISWC_RDFaEditor/public.pdf`
12. Morbidoni, C.: SWickyNotes Starting Guide. Net7 and Universita Politecnica delle Marche (April 2012), `http://www.swickynotes.org/docs/SWickyNotesStartingGuide.pdf`
13. Dello, K., Simperl, E.P.B., Tolksdorf, R.: Creating and using semantic web information with makna. In: First Workshop on Semantic Wikis - From Wiki to Semantics (2006)
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. Schaffert, S.: Ikewiki: A semantic wiki for collaborative knowledge management. In: Workshop on Semantic Technologies in Collaborative Applications, Manchester, UK (June 2006)
16. Hinze, A., Voisard, A., Buchanan, G.: Tip: Personalizing information delivery in a tourist information system. J. of IT & Tourism 11(3), 247–264 (2009)

17. Kruk, S.R., Woroniecki, T., Gzella, A., Dabrowski, M.: JeromeDL - a semantic digital library. In: Semantic Web Challenge (2007)
18. faviki: faviki - tags that make sense, http://www.faviki.com
19. Passant, A.: MOAT-project, http://moat-project.org
20. Teufel, S.: Argumentative Zoning: Information Extraction from Scientific Text. PhD thesis, University of Edinburgh (1999)
21. Handschuh, S.: Creating Ontology-based Metadata by Annotation for the Semantic Web. Ph.D. thesis (dr. rer. pol.), University of Karlsruhe, TH (2005)
22. Bayerl, P.S., Lüngen, H., Gut, U., Paul, K.I.: Methodology for reliable schema development and evaluation of manual annotations. In: Workshop on Knowledge Markup and Semantic Annotation, pp. 17–23 (2003)
23. Erdmann, M., Maedche, A., Schnurr, H., Staab, S.: From manual to semi-automatic semantic annotation: About ontology-based text annotation tools. Group 6(i), 79–91 (2000)
24. Maynard, D., Dasiopoulou, S., et al.: D1.2.2.1.3 Benchmarking of annotation tools. Technical report, Knowledge Web Project (2007)
25. Castro, R.: Benchmarking Semantic Web technology. Studies on the Semantic Web. IOS Press (2010)
26. Degler, D., Henninger, S., Battle, L.: Semantic Web HCI: discussing research implications. In: Extended Abstracts on Human Factors in Computing Systems, pp. 1909–1912. ACM (2007)
27. Schraefel, M., Karger, D.: The pathetic fallacy of rdf. In: International Workshop on the Semantic Web and User Interaction (SWUI 2006) (2006)
28. Uren, V., Cimiano, P., et al.: Semantic annotation for knowledge management: Requirements and a survey of the state of the art. Web Semant. 4(1), 14–28 (2006)
29. Karger, D.: Unference: Ui (not ai) as key to the semantic web. Panel on Interaction Design Grand Challenges and the Semantic Web at Semantic Web User Interaction Workshop (2006), http://swui.semanticweb.org/swui06/panel/Karger.ppt
30. Jameson, A.: Usability and the Semantic Web. In: Sure, Y., Domingue, J. (eds.) ESWC 2006. LNCS, vol. 4011, p. 3. Springer, Heidelberg (2006)
31. Duineveld, A.J., Stoter, R., et al.: Wondertools? a comparative study of ontological engineering tools. Journal of Human-Computer Studies 52(6), 1111–1133 (2000)
32. Flouris, G., Manakanatas, D., Kondylakis, H., Plexousakis, D., Antoniou, G.: Ontology change: Classification and survey. Knowl. Eng. Rev. 23(2), 117–152 (2008)
33. Maedche, A., Staab, S.: Ontology learning for the semantic web. IEEE Intelligent Systems 16(2), 72–79 (2001)
34. Jones, W., Karger, D., Bergman, O., Franklin, M., Pratt, W., Bates, M.: Towards a Unification & Integration of PIM support. Technical report, University of Washington (2005)
35. Aimé, X., Furst, F., Kuntz, P., Trichet, F.: Conceptual and Lexical Prototypicality Gradients Dedicated to Ontology Personalisation. In: Meersman, R., Tari, Z. (eds.) OTM 2008, Part II. LNCS, vol. 5332, pp. 1423–1439. Springer, Heidelberg (2008)
36. Hogben, G., Wilikens, M., Vakalis, I.: On the Ontology of Digital Identification. In: Meersman, R. (ed.) OTM Workshops 2003. LNCS, vol. 2889, pp. 579–593. Springer, Heidelberg (2003)
37. Handschuh, S., Staab, S.: Authoring and annotation of web pages in cream. In: WWW, pp. 462–473 (2002)
38. Heese, R., Luczak-Rösch, M., Paschke, A., Oldakowski, R., Streibel, O.: One click annotation. In: Workshop on Scripting and Development for the Semantic Web (May 2010)
39. Degler, D.: Design 10:5:2 for semantic applications. In: Semantic Technology Conference (2011), http://www.designforsemanticweb.com/

40. Snyder, C.: Paper prototyping: The fast and easy way to design and refine user interfaces. Morgan Kaufmann Pub. (2003)
41. IBM: User expectations (2012), http://www-01.ibm.com/software/ucd/initial/expectations.html
42. Schlegel, A., Heese, R., Hinze, A.: Visualisation of semantic enrichment. In: Interaction and Visualisation in the Data Web, Workshop at Informatik 2012 (2012)
43. Hinze, A., Heese, R., Schlegel, A., Luczak-Rösch, M.: User-Defined Semantic Enrichment of Full-Text Documents: Experiences and Lessons Learned. In: Zaphiris, P., Buchanan, G., Rasmussen, E., Loizides, F. (eds.) TPDL 2012. LNCS, vol. 7489, pp. 209–214. Springer, Heidelberg (2012)

Ontology-Based Access to Probabilistic Data with OWL QL

Jean Christoph Jung and Carsten Lutz

Universität Bremen, Germany
{jeanjung,clu}@informatik.uni-bremen.de

Abstract. We propose a framework for querying probabilistic instance data in the presence of an OWL2 QL ontology, arguing that the interplay of probabilities and ontologies is fruitful in many applications such as managing data that was extracted from the web. The prime inference problem is computing answer probabilities, and it can be implemented using standard probabilistic database systems. We establish a PTime vs. #P dichotomy for the data complexity of this problem by lifting a corresponding result from probabilistic databases. We also demonstrate that query rewriting (backwards chaining) is an important tool for our framework, show that non-existence of a rewriting into first-order logic implies #P-hardness, and briefly discuss approximation of answer probabilities.

1 Introduction

There are many applications that require data to be first extracted from the web and then further processed locally, by feeding it into a relational database system (RDBMS). Such web data differs in several crucial aspects from traditional data stored in RDBMSs: on the one hand, it is *uncertain* because of the unreliability of many web data sources and due to the data extraction process, which relies on heuristic decisions and is significantly error prone [23]; on the other hand, web data is often provided without explicit schema information and thus requires *interpretation* based on ontologies and other semantic techniques. This latter aspect is addressed by ontology languages such as OWL2. In particular, the OWL2 QL profile is a popular lightweight ontology language that is tailored towards enriching standard RDBMS query answering with an ontology component, thus allowing the user of semantic technologies to take advantage of the maturity and effiency of such systems [6]. While the current techniques developed around OWL2 QL are well-suited to deal with the interpretation aspect of web data, they are not able to deal with its uncertainty. In this paper, we propose and analyze a framework for data storage and querying that supports ontologies formulated in (a fragment of) OWL2 QL, but also features prominent aspects of probabilistic database models to explicitly represent uncertainty. In a nutshell, our approach relates to probabilistic database systems (PDBMSs) in the same way that traditional OWL2 QL query answering relates to RDBMSs.

In our framework, we adopt data models from the recently very active area of probabilistic databases [7,31], but use an open world assumption as is standard in the context of OWL2 QL. Specifically, we store data in description logic ABoxes enriched with probabilities that are attached to *probabilistic events*, which can either be modeled explicitly (resulting in what we call *pABoxes*) or be implicitly associated with each ABox assertion (resulting in *ipABoxes*). For example, a pABox assertion SoccerPlayer(messi) can be associated with an *event expression* $e_1 \vee e_2$, where e_1 and e_2 represent events such as 'web extraction tool x correctly analyzed webpage y stating that Messi is a soccer player'. We generally assume all events to be probabilistically independent, which results in a straightforward semantics that is similar to well-known probabilistic versions of datalog [27,12]. Ontologies are represented by TBoxes formulated in the description logic DL-Lite, which forms a logical core of the ontology language OWL2 QL. We are then interested in *computing the answer probabilities* to *conjunctive queries (CQs)*; note that probabilities can occur only in the data, but neither in the ontology nor in the query. We believe that this setup is of general interest and potentially useful for a wide range of applications including the management of data extracted from the web, machine translation, and dealing with data that arises from sensor networks. All these applications can potentially benefit from a fruitful interplay between ontologies and probabilities; in particular, we argue that the ontology can help to reduce the uncertainty of the data.

In database research, practical feasibility is usually identified with PTime data complexity, where *data complexity* means to treat only the (probabilistic) data as an input while considering both the TBox and the query to be fixed. The main aim of this paper is to study the data complexity of *ontology-based access to probabilistic data (pOBDA)* in the concrete framework described above. As a central tool, we use *query rewriting* (also called *backwards chaining*), which is an important technique for traditional *ontology based data access (OBDA)*, i.e., answering CQs in the presence of a DL-Lite TBox over non-probabilistic data [6]. Specifically, the idea is to rewrite a given CQ q and DL-Lite TBox $\mathcal{T}$ into an SQL (equivalently: first-order) query $q_\mathcal{T}$ such that for any ABox $\mathcal{A}$, the certain answers to q over $\mathcal{A}$ relative to $\mathcal{T}$ are identical with the answers to $q_\mathcal{T}$ over $\mathcal{A}$ viewed as a relational database instance. We set out with observing that rewritings from traditional OBDA are useful also in the context of pOBDA: for any pABox $\mathcal{A}$, the probability that a tuple $\boldsymbol{a}$ is a certain answer to q over $\mathcal{A}$ relative to $\mathcal{T}$ is identical to the probability that $\boldsymbol{a}$ is an answer to $q_\mathcal{T}$ over $\mathcal{A}$ viewed as a probabilistic database. This *lifting* of query rewriting to the probabilistic case immediately implies that one can implement pOBDA based on existing PDBMSs such as MayBMS, Trio, and MystiQ [1,33,5].

Lifting also allows us to carry over the dichotomy between PTime and #P-hardness for computing the probabilities of answers to unions of conjunctive queries (UCQs) over probabilistic databases recently obtained by Dalvi, Suciu, and Schnaitter [8] to our pOBDA framework provided that we restrict ourselves to ipABoxes, which are strictly less expressive than pABoxes. Based on a careful syntactic analysis, we provide a transparent characterization of those CQs q and DL-Lite TBoxes $\mathcal{T}$ for which computing answer probabilities is in PTime. We then proceed to showing that query rewriting is a *complete* tool for proving

PTime data complexity in pOBDA, in the following sense: we replace DL-Lite with the strictly more expressive description logic $\mathcal{ELI}$, which is closely related to the OWL2 profile OWL2 EL and where, in contrast to DL-Lite, rewritings into first-order queries do not exist for every CQ q and TBox $\mathcal{T}$; we then prove that if any $(q, \mathcal{T})$ does *not* have a rewriting, then computing answer probabilities for q relative to $\mathcal{T}$ is #P-hard. Thus, if it is possible at all to prove that some $(q, \mathcal{T})$ has PTime data complexity, then this can always be done using query rewriting. Both in DL-Lite and $\mathcal{ELI}$, the class of queries and TBoxes with PTime data complexity is relatively small, which leads us to also consider the approximation of answer probabilities, based on the notion of a *fully polynomial randomized approximation scheme (FPRAS).* It is not hard to see that all pairs $(q, \mathcal{T})$ have an FPRAS when $\mathcal{T}$ is formulated in DL-Lite, but this is not the case for more expressive ontology languages such as $\mathcal{ALC}$. Note that these results are in the spirit of the *non-uniform* analysis of data complexity recently initiated in an OBDA context in [26]. We defer some proofs to the appendix of the long version of this paper, available at http://www.informatik.uni-bremen.de/tdki/research/papers.html.

Related Work. The probabilistic ABox formalism studied in this paper is inspired by the probabilistic database models in [9], but can also be viewed as a variation of probabilistic versions of datalog and Prolog, see [27,12] and references therein. There have recently been other approaches to combining ontologies and probabilities for data access [11,14], with a different semantics; the setup considered by Gottlob, Lukasiewicz, and Simari in [14] is close in spirit to the framework studied here, but also allows probabilities in the TBox and has a different, rather intricate semantics based on Markov logic. In fact, we deliberately avoid probabilities in the ontology because (i) this results in a simple and fundamental, yet useful formalism that still admits a very transparent semantics and (ii) it enables the use of standard PDBMSs for query answering. There has also been a large number of proposals for enriching description logic TBoxes (instead of ABoxes) with probabilities, see [24,25] and the references therein. Our running application example is web data extraction, in the spirit of [16] to store extracted web data in a probabilistic database. Note that It has also been proposed to integrate both probabilities and ontologies directly into the data extraction tool [13]. We believe that both approaches can be useful and could even be orchestrated to play together. Finally, we note that the motivation for our framework is somewhat similar to what is done in [30], where the retrieval of top-k-answers in OBDA is considered under a fuzzy logic-like semantics based on 'scoring functions'.

2 Preliminaries

We use standard notation for the syntax and semantics of description logics (DLs) and refer to [3] for full details. As usual, $\mathsf{N_C}$, $\mathsf{N_R}$, and $\mathsf{N_I}$ denote countably infinite sets of concept names, role names, and individual names, C, D denote (potentially) composite concepts, A, B concept names, r, s role names, R and S

role names or their inverse, and a, b individual names. When $R = r^-$, then as usual R^- denotes r. We consider the following DLs.

In *DL-Lite*, *TBoxes* are finite sets of *concept inclusions (CIs)* $B \sqsubseteq B'$ and $B \sqcap B' \sqsubseteq \bot$ with B and B' concepts of the form $\exists r$, $\exists r^-$, $\top$ or A. Note that there is no nesting of concept constructors in DL-Lite. This version is sometimes called *DL-Lite*$_{\mathsf{core}}$ and includes crucial parts of the OWL2 QL profile; some features of OWL2 QL are omitted in DL-Lite$_{\mathsf{core}}$, mainly to keep the presentation simple.

$\mathcal{ELI}$ is a generalization of DL-Lite without $\bot$ (which we will largely disregard in this paper for reasons explained later on) and offers the concept constructors $\top$, $C \sqcap D$, $\exists r.C$, and $\exists r^-.C$. In $\mathcal{ELI}$, a *TBox* is a finite set of CIs $C \sqsubseteq D$ with C and D (potentially) compound concepts.

In DLs, data is stored in an *ABox*, which is a finite set of *concept assertions* $A(a)$ and *role assertions* $r(a, b)$. We use $\mathsf{Ind}(\mathcal{A})$ to denote the set of individual names used in the ABox $\mathcal{A}$ and sometimes write $r^-(a, b) \in \mathcal{A}$ for $r(b, a) \in \mathcal{A}$.

The semantics of DLs is based on interpretations $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$ as usual [3]. An interpretation is a *model* of a TBox $\mathcal{T}$ (resp. ABox $\mathcal{A}$) if it satisfies all concept inclusions in $\mathcal{T}$ (resp. assertions in $\mathcal{A}$), where satisfaction is defined in the standard way. An ABox $\mathcal{A}$ is *consistent* w.r.t. a TBox $\mathcal{T}$ if $\mathcal{A}$ and $\mathcal{T}$ have a common model. We write $\mathcal{T} \models C \sqsubseteq D$ if for all models $\mathcal{I}$ of $\mathcal{T}$, $C^\mathcal{I} \subseteq D^\mathcal{I}$ and say that *C is subsumed by D* relative to $\mathcal{T}$.

Conjunctive queries (CQs) take the form $\exists \boldsymbol{y}.\varphi(\boldsymbol{x}, \boldsymbol{y})$, with φ a conjunction of atoms of the form $A(t)$ and $r(t, t')$ and where $\boldsymbol{x}, \boldsymbol{y}$ denote (tuples of) variables taken from a set $\mathsf{N_V}$ and t, t' denote *terms*, i.e., a variable or an individual name. We call the variables in $\boldsymbol{x}$ the *answer variables* and those in $\boldsymbol{y}$ the *quantified* variables. The set of all variables in a CQ q is denoted by $\mathsf{var}(q)$ and the set of all terms in q by $\mathsf{term}(q)$. A CQ q is *n-ary* if it has n answer variables and *Boolean* if it is 0-ary. Whenever convenient, we treat a CQ as a *set* of atoms and sometimes write $r^-(t, t') \in q$ instead of $r(t', t) \in q$.

Let $\mathcal{I}$ be an interpretation and q a CQ with answer variables $x_1, \ldots, x_k$. For $\boldsymbol{a} = a_1 \cdots a_k \in \mathsf{N_I}^k$, an *$\boldsymbol{a}$-match* for q in $\mathcal{I}$ is a mapping $\pi : \mathsf{term}(q) \to \Delta^\mathcal{I}$ such that $\pi(x_i) = a_i$ for $1 \leq i \leq k$, $\pi(a) = a^\mathcal{I}$ for all $a \in \mathsf{term}(q) \cap \mathsf{N_I}$, $\pi(t) \in A^\mathcal{I}$ for all $A(t) \in q$, and $(\pi(t_1), \pi(t_2)) \in r^\mathcal{I}$ for all $r(t_1, t_2) \in q$. We write $\mathcal{I} \models q[\boldsymbol{a}]$ if there is an $\boldsymbol{a}$-match of q in $\mathcal{I}$. For a TBox $\mathcal{T}$ and an ABox $\mathcal{A}$, we write $\mathcal{T}, \mathcal{A} \models q[\boldsymbol{a}]$ if $\mathcal{I} \models q[\boldsymbol{a}]$ for all models $\mathcal{I}$ of $\mathcal{T}$ and $\mathcal{A}$. In this case and when all elements of $\boldsymbol{a}$ are from $\mathsf{Ind}(\mathcal{A})$, $\boldsymbol{a}$ is a *certain answer* to q w.r.t. $\mathcal{A}$ and $\mathcal{T}$. We use $\mathsf{cert}_\mathcal{T}(q, \mathcal{A})$ to denote the set of all certain answers to q w.r.t. $\mathcal{A}$ and $\mathcal{T}$.

As done often in the context of OBDA, we adopt the unique name assumption (UNA), which requires that $a^\mathcal{I} \neq b^\mathcal{I}$ for all interpretations $\mathcal{I}$ and all $a, b \in \mathsf{N_I}$ with $a \neq b$. Note that, in all logics studied here, query answers with and without UNA actually coincide, so the assumption of the UNA is without loss of generality.

3 Probabilistic OBDA

We introduce our framework for probabilistic OBDA, starting with the definition of a rather general, probabilistic version of ABoxes. Let $\mathcal{E}$ be a countably infinite set of *atomic (probabilistic) events*. An *event expression* is built up from atomic events using the Boolean operators $\neg$, $\wedge$, $\vee$. We use $\mathsf{expr}(\mathcal{E})$ to denote the set of all event expressions over $\mathcal{E}$. A *probability assignment for E* is a map $E \to [0,1]$.

Definition 1 (pABox). *A* probabilistic ABox (pABox) *is of the form $(\mathcal{A}, e, p)$ with $\mathcal{A}$ an ABox, e a map $\mathcal{A} \to \mathsf{expr}(\mathcal{E})$, and p a probability assignment for $E_\mathcal{A}$, the atomic events in $\mathcal{A}$.*

Example 1. We consider as a running example a (fictitious) information extraction tool that is gathering data from the web, see [16] for a similar setup. Assume we are gathering data about soccer players and the clubs they play for in the current 2012 season, and we want to represent the result as a pABox.

(1) The tool processes a newspaper article stating that 'Messi is the soul of the Argentinian national soccer team'. Because the exact meaning of this phrase is unclear (it could refer to a soccer player, a coach, a mascot), it generates the assertion $\mathsf{Player}(\mathsf{messi})$ associated with the atomic event expression e_1 with $p(e_1) = 0.7$. The event e_1 represents that the phrase was interpreted correctly.

(2) The tool finds the Wikipedia page on Lionel Messi, which states that he is a soccer player. Since Wikipedia is typically reliable and up to date, but not *always* correct, it updates the expression associated with $\mathsf{Player}(\mathsf{messi})$ to $e_1 \vee e_2$ and associates e_2 with $p(e_2) = 0.95$.

(3) The tool finds an HTML table on the homepage of FC Barcelona saying that the top scorers of the season are Messi, Villa, and Pedro. It is not stated whether the table refers to the 2011 or the 2012 season, and consequently we generate the ABox assertions $\mathsf{playsfor}(x, \mathsf{FCbarca})$ for $x \in \{\mathsf{messi}, \mathsf{villa}, \mathsf{pedro}\}$ all associated with the same atomic event expression e_3 with $p(e_3) = 0.5$. Intuitively, the event e_3 is that the table refers to 2012.

(4) Still processing the table, the tool applies the background knowledge that top scorers are typically strikers. It generates three assertions $\mathsf{Striker}(x)$ with $x \in \{\mathsf{messi}, \mathsf{villa}, \mathsf{pedro}\}$, associated with atomic events e_4, e_4', and e_4''. It sets $p(e_4) = p(e_4') = p(e_4'') = 0.8$. The probability is higher than in (3) since being a striker is a more stable property than playing for a certain club, thus this information does not depend so much on whether the table is from 2011 or 2012.

(5) The tool processes the twitter message 'Villa was the only one to score a goal in the match between Barca and Real'. It infers that Villa plays either for Barcelona or for Madrid, generating the assertions $\mathsf{playsfor}(\mathsf{villa}, \mathsf{FCbarca})$ and $\mathsf{playsfor}(\mathsf{villa}, \mathsf{realmadrid})$. The first assertion is associated with the event e_5, the second one with $\neg e_5$. It sets $p(e_5) = 0.5$.

Now for the semantics of pABoxes $(\mathcal{A}, e, p)$. Each $E \subseteq E_\mathcal{A}$ can be viewed as a truth assignment that makes all events in E true and all events in $E_\mathcal{A} \setminus E$ false.

Definition 2. *Let $(\mathcal{A}, e, p)$ be a pABox. For each $E \subseteq E_\mathcal{A}$, define a corresponding non-probabilistic ABox $\mathcal{A}_E := \{\alpha \in \mathcal{A} \mid E \models e(\alpha)\}$. The function p represents a probability distribution on $2^{E_\mathcal{A}}$, by setting for each $E \subseteq E_\mathcal{A}$:*

$$p(E) = \prod_{e \in E} p(e) \cdot \prod_{e \in E_\mathcal{A} \setminus E} (1 - p(e)).$$

The probability of an answer $\boldsymbol{a} \in \mathsf{Ind}(\mathcal{A})^n$ *to an n-ary conjunctive query q over a pABox $\mathcal{A}$ and TBox $\mathcal{T}$ is*

$$p_{\mathcal{A},\mathcal{T}}(\boldsymbol{a} \in q) = \sum_{E \subseteq E_{\mathcal{A}}\,:\,\boldsymbol{a} \in \mathsf{cert}_{\mathcal{T}}(q,\mathcal{A}_E)} p(E).$$

For Boolean CQs q, we write $p(\mathcal{A},\mathcal{T} \models q)$ instead of $p_{\mathcal{A},\mathcal{T}}(() \in q)$, where $()$ denotes the empty tuple.

Example 2. Consider again the web data extraction example discussed above. To illustrate how ontologies can help to reduce uncertainty, we use the DL-Lite TBox

$$\begin{aligned} \mathcal{T} = \{ \quad & \exists \mathsf{playsfor} \sqsubseteq \mathsf{Player} & \mathsf{Player} \sqsubseteq \exists \mathsf{playsfor} \\ & \exists \mathsf{playsfor}^- \sqsubseteq \mathsf{SoccerClub} & \mathsf{Striker} \sqsubseteq \mathsf{Player} \quad \} \end{aligned}$$

Consider the following subcases considered above.

(1) + (3) The resulting pABox comprises the following assertions with associated event expressions:

$$\begin{array}{rr} \mathsf{Player}(\mathsf{messi}) \rightsquigarrow e_1 & \mathsf{playsfor}(\mathsf{messi}, \mathsf{FCbarca}) \rightsquigarrow e_3 \\ \mathsf{playsfor}(\mathsf{villa}, \mathsf{FCbarca}) \rightsquigarrow e_3 & \mathsf{playsfor}(\mathsf{pedro}, \mathsf{FCbarca}) \rightsquigarrow e_3 \end{array}$$

with $p(e_1) = 0.7$ and $p(e_3) = 0.5$. Because of the statement $\exists \mathsf{playsfor} \sqsubseteq \mathsf{Player}$, using $\mathcal{T}$ (instead of an empty TBox) increases the probability of messi to be an answer to the query $\mathsf{Player}(x)$ from 0.7 to 0.85.

(5) The resulting pABox is

$$\mathsf{playsfor}(\mathsf{villa}, \mathsf{FCbarca}) \rightsquigarrow e_5 \quad \mathsf{playsfor}(\mathsf{villa}, \mathsf{realmadrid}) \rightsquigarrow \neg e_5$$

with $p(e_5) = 0.5$. Although $\mathsf{Player}(\mathsf{villa})$ does not occur in the data, the probability of villa to be an answer to the query $\mathsf{Player}(x)$ is 1 (again by the TBox-statement $\exists \mathsf{playsfor} \sqsubseteq \mathsf{Player}$).

(3)+(4) This results in the pABox

$$\begin{array}{rr} \mathsf{playsfor}(\mathsf{messi}, \mathsf{FCbarca}) \rightsquigarrow e_3 & \mathsf{Striker}(\mathsf{messi}) \rightsquigarrow e_4 \\ \mathsf{playsfor}(\mathsf{villa}, \mathsf{FCbarca}) \rightsquigarrow e_3 & \mathsf{Striker}(\mathsf{villa}) \rightsquigarrow e_4' \\ \mathsf{playsfor}(\mathsf{pedro}, \mathsf{FCbarca}) \rightsquigarrow e_3 & \mathsf{Striker}(\mathsf{pedro}) \rightsquigarrow e_4'' \end{array}$$

with $p(e_3) = 0.5$ and $p(e_4) = p(e_4') = p(e_4'') = 0.8$. Due to the last three CIs in $\mathcal{T}$, each of messi, villa, pedro is an answer to the CQ $\exists y.\mathsf{playsfor}(x, y) \wedge \mathsf{SoccerClub}(y)$ with probability 0.9.

Related Models in Probabilistic Databases. Our pABoxes can be viewed as an open world version of the probabilistic data model studied by Dalvi and Suciu in [9]. It is as a less succinct version of *c-tables*, a traditional data model for probabilistic databases due to Imielinski and Lipski [18]. Nowadays, there is an abundance of probabilistic data models, see [15,29,2] and the references therein.

All these models provide a compact representation of distributions over potentially large sets of *possible worlds*. Since we are working with an open world semantics, pABoxes instead represent a distribution over *possible world descriptions*. Each such description may have any number of models. Note that our semantics is similar to the semantics of ("type 2") probabilistic first-order and description logics [17,25].

Dealing with Inconsistencies. Of course, some of the ABoxes $\mathcal{A}_E$ might be inconsistent w.r.t. the TBox $\mathcal{T}$ used. In this case, it may be undesirable to let them contribute to the probabilities of answers. For example, if we use the pABox

$$\mathsf{Striker}(\mathsf{messi}) \rightsquigarrow e_1 \quad \mathsf{Goalie}(\mathsf{messi}) \rightsquigarrow e_2$$

with $p(e_1) = 0.8$ and $p(e_2) = 0.3$ and the TBox $\mathsf{Goalie} \sqcap \mathsf{Striker} \sqsubseteq \bot$, then messi is an answer to the query $\mathsf{SoccerClub}(x)$ with probability 0.24 while one would probably expect it to be zero (which is the result when the empty TBox is used). We follow Antova, Koch, and Olteanu and advocate a pragmatic solution based on *rescaling* [2]. More specifically, we remove those ABoxes $\mathcal{A}_E$ that are inconsistent w.r.t. $\mathcal{T}$ and rescale the remaining set of ABoxes so that they sum up to probability one. In other words, we set

$$\widehat{p}_{\mathcal{A},\mathcal{T}}(\boldsymbol{a} \in q) = \frac{p_{\mathcal{A},\mathcal{T}}(\boldsymbol{a} \in q) - p(\mathcal{A},\mathcal{T} \models \bot)}{1 - p(\mathcal{A},\mathcal{T} \models \bot)}$$

where $\bot$ is a Boolean query that is entailed exactly by those ABoxes $\mathcal{A}$ that are inconsistent w.r.t. $\mathcal{T}$. The rescaled probability $\widehat{p}_{\mathcal{A},\mathcal{T}}(\boldsymbol{a} \in q)$ can be computed in PTime when this is the case both for $p_{\mathcal{A},\mathcal{T}}(\boldsymbol{a} \in q)$ and $p(\mathcal{A},\mathcal{T} \models \bot)$. Note that rescaling results in some effects that might be unexpected such as reducing the probability of messi to be an answer to $\mathsf{Striker}(x)$ from 0.8 to ≈0.74 when the above TBox is added.

In the remainder of the paper, for simplicity we will only admit TBoxes $\mathcal{T}$ such that all ABoxes $\mathcal{A}$ are consistent w.r.t. $\mathcal{T}$.

4 Query Rewriting

The main computational problem in traditional OBDA is, given an ABox $\mathcal{A}$, query q, and TBox $\mathcal{T}$, to produce the certain answers to q w.r.t. $\mathcal{A}$ and $\mathcal{T}$. In the context of lightweight DLs such as DL-Lite, a prominent approach to address this problem is to use *FO-rewriting*, which yields a reduction to query answering in relational databases. The aim of this section is to show that this approach is fruitful also in the case of computing answer probabilities in probabilistic OBDA. In particular, we use it to lift the PTime vs. #P dichotomy result on probabilistic databases recently obtained by Dalvi, Suciu, and Schnaitter [8] to probabilistic OBDA in DL-Lite.

4.1 Lifting FO-Rewritings to Probabilistic OBDA

We first describe the query rewriting approach to traditional OBDA. A *first-order query (FOQ)* is a first-order formula $q(\boldsymbol{x})$ constructed from atoms $A(x)$ and $r(x, y)$ using negation, conjunction, disjunction, and existential quantification. The free variables $\boldsymbol{x}$ are the *answer variables* of $q(\boldsymbol{x})$. For an interpretation $\mathcal{I}$, we write $\mathsf{ans}(q, \mathcal{I})$ to denote the *answers to* q *in* $\mathcal{I}$, i.e., the set of all tuples $\boldsymbol{a}$ such that $\mathcal{I} \models q[\boldsymbol{a}]$. In what follows, we use $\mathcal{I}_\mathcal{A}$ to denote the ABox $\mathcal{A}$ viewed as an interpretation (in the obvious way). A *first-order (FO) TBox* is a finite set of first-order sentences.

Definition 3 (FO-rewritable). *A CQ q is* FO-rewritable relative to an FO TBox $\mathcal{T}$ *if one can effectively construct a FOQ $q_\mathcal{T}$ such that* $\mathsf{cert}_\mathcal{T}(q, \mathcal{A}) = \mathsf{ans}(q_\mathcal{T}, \mathcal{I}_\mathcal{A})$ *for every ABox $\mathcal{A}$. In this case, $q_\mathcal{T}$ is a* rewriting *of q relative to $\mathcal{T}$.*

For computing the answers to q w.r.t. $\mathcal{A}$ and $\mathcal{T}$ in traditional OBDA, one can thus construct $q_\mathcal{T}$ and then hand it over for execution to a database system that stores $\mathcal{A}$.

The following observation states that FO-rewritings from traditional OBDA are also useful in probabilistic OBDA. We use $p^d_\mathcal{A}(\boldsymbol{a} \in q)$ to denote the probability that $\boldsymbol{a}$ is an answer to the query q given the pABox $\mathcal{A}$ viewed as a probabilistic database in the sense of Dalvi and Suciu [8]. More specifically,

$$p^d_\mathcal{A}(\boldsymbol{a} \in q) = \sum_{E \subseteq E_\mathcal{A} \mid \boldsymbol{a} \in \mathsf{ans}(q, \mathcal{I}_{\mathcal{A}_E})} p(E)$$

The following is immediate from the definitions.

Theorem 1 (Lifting). *Let $\mathcal{T}$ be an FO TBox, $\mathcal{A}$ a pABox, q an n-ary CQ, $\boldsymbol{a} \in \mathsf{Ind}(\mathcal{A})^n$ a candidate answer for q, and $q_\mathcal{T}$ an FO-rewriting of q relative to $\mathcal{T}$. Then $p_{\mathcal{A},\mathcal{T}}(\boldsymbol{a} \in q) = p^d_\mathcal{A}(\boldsymbol{a} \in q_\mathcal{T})$.*

From an application perspective, Theorem 1 enables the use of probabilistic database systems such as MayBMS, Trio, and MystiQ for implementing probabilistic OBDA [1,33,5]. Note that it might be necessary to adapt pABoxes in an appropriate way in order to match the data models of these systems. However, such modifications do not impair applicability of Theorem 1.

From a theoretical viewpoint, Theorem 1 establishes query rewriting as a useful tool for analyzing data complexity in probabilistic OBDA. We say that a CQ q *is in* PTIME *relative to a TBox* $\mathcal{T}$ if there is a polytime algorithm that, given an ABox $\mathcal{A}$ and a candidate answer $\boldsymbol{a} \in \mathsf{Ind}(\mathcal{A})^n$ to q, computes $p_{\mathcal{A},\mathcal{T}}(\boldsymbol{a} \in q)$. We say that q is #P*-hard relative to* $\mathcal{T}$ if the afore mentioned problem is hard for the counting complexity class #P, please see [32] for more information. We pursue a non-uniform approach to the complexity of query answering in probabilistic OBDA, as recently initiated in [26]: ideally, we would like to understand the precise complexity of every CQ q relative to every TBox $\mathcal{T}$, against the background of some preferably expressive 'master logic' used for $\mathcal{T}$. Note, though, that our framework yields one counting problem for each CQ and

TBox, while [26] has one decision problem for each TBox, quantifying over all CQs.

Unsurprisingly, pABoxes are too strong a formalism to admit *any* tractable queries worth mentioning. An n-ary CQ q is *trivial* for a TBox $\mathcal{T}$ iff for every ABox $\mathcal{A}$, we have $\mathsf{cert}_{\mathcal{T}}(\mathcal{A}, q) = \mathsf{Ind}(\mathcal{A})^n$.

Theorem 2. *Over pABoxes, every CQ q is #P-hard relative to every first-order TBox $\mathcal{T}$ for which it is nontrivial.*

Proof. The proof is by reduction of counting the number of satisfying assignments of a propositional formula.[1] Assume that q has answer variables $x_1, \ldots, x_n$ and let φ be a propositional formula over variables $z_1, \ldots, z_m$. Convert φ into a pABox $\mathcal{A}$ as follows: take q viewed as an ABox, replacing every variable x with an individual name a_x; then associate every ABox assertion with φ viewed as an event expression over events $z_1, \ldots, z_m$ and set $p(z_i) = 0.5$ for all i. We are interested in the answer $\boldsymbol{a} = a_{x_1} \cdots a_{x_n}$. For all $E \subseteq E_{\mathcal{A}}$ with $E \not\models \varphi$, we have $\mathcal{A}_E = \emptyset$ and thus $\boldsymbol{a} \notin \mathsf{cert}_{\mathcal{T}}(q, \mathcal{A}_E)$ since q is non-trivial for $\mathcal{T}$. For all $E \subseteq E_{\mathcal{A}}$ with $E \models \varphi$, the ABox $\mathcal{A}_E$ is the ABox-representation of q and thus $\boldsymbol{a} \in \mathsf{cert}_{\mathcal{T}}(q, \mathcal{A}_E)$. Consequently, the number of assignments that satisfy φ is $p_{\mathcal{A},\mathcal{T}}(\overline{a} \in q) * 2^m$. Thus, there is a PTime algorithm for counting the number of satisfying assignments given an oracle for computing answer probabilities for q and $\mathcal{T}$. ❑

Theorem 2 motivates the study of more lightweight probabilistic ABox formalisms. While pABoxes (roughly) correspond to c-tables, which are among the most expressive probabilistic data models, we now move to the other end of the spectrum and introduce ipABoxes as a counterpart of *tuple independent databases* [9,12]. Argueably, the latter are the most inexpressive probabilistic data model that is still useful.

Definition 4 (ipABox). *An* assertion-independent probabilistic ABox (short: ipABox) *is a probabilistic ABox in which all event expressions are atomic and where each atomic event expression is associated with at most one ABox assertion.*

To save notation, we write ipABoxes in the form $(\mathcal{A}, p)$ where $\mathcal{A}$ is an ABox and p is a map $\mathcal{A} \to [0,1]$ that assigns a probability to each ABox assertion. In this representation, the events are only implicit (one atomic event per ABox assertion). For $\mathcal{A}' \subseteq \mathcal{A}$, we write $p(\mathcal{A}')$ as a shorthand for $p(\{e \in E \mid \exists \alpha \in \mathcal{A}' : e(\alpha) = e\})$. Note that $p(\mathcal{A}') = \prod_{\alpha \in \mathcal{A}'} p(\alpha) \cdot \prod_{\alpha \in \mathcal{A} \setminus \mathcal{A}'}(1 - p(\alpha))$ and thus all assertions in an ipABox can be viewed as independent events; also note that for any CQ q, we have $p_{\mathcal{A},\mathcal{T}}(\boldsymbol{a} \in q) = \sum_{\mathcal{A}' \subseteq \mathcal{A} : \boldsymbol{a} \in \mathsf{cert}_{\mathcal{T}}(q, \mathcal{A}')} p(\mathcal{A}')$. Cases (1) and (4) of our web data extraction example yield ipABoxes, whereas cases (2), (3), and (5) do not. We refer to [31] for a discussion of the usefulness of ipABoxes/tuple independent databases. For the remainder of the paper, we assume that only ipABoxes are admitted unless explicitly noted otherwise.

[1] Throughout the paper, we use the standard oracle-based notion of reduction originally introduced by Valiant in the context of counting complexity [32].

4.2 Lifting the PTime vs. #P Dichotomy

We now use Theorem 1 to lift a PTime vs. #P dichotomy recently obtained in the area of probabilistic databases to probabilistic OBDA in DL-Lite. Note that, for any CQ and DL-Lite TBox, an FO-rewriting is guaranteed to exist [6]. The central observation is that, by Theorem 1, computing the probability of answers to a CQ q relative to a TBox $\mathcal{T}$ over ipABoxes is *exactly the same problem* as computing the probability of answers to $q_{\mathcal{T}}$ over (ipABoxes viewed as) tuple independent databases. We can thus analyze the complexity of CQs/TBoxes over ipABoxes by analyzing the complexity of their rewritings. In particular, standard rewriting techniques produce for each CQ and DL-Lite TBox an FO-rewriting that is a union of conjunctive queries (a UCQ) and thus, together with Theorem 1, Dalvi, Suciu and Schnaitter's PTime vs. #P dichotomy for UCQs over tuple independent databases [8] immediately yields the following.

Theorem 3 (Abstract Dichotomy). *Let q be a CQ and $\mathcal{T}$ a DL-Lite TBox. Then q is in* PTime *relative to $\mathcal{T}$ or q is #*P*-hard relative to $\mathcal{T}$.*

Note that Theorem 3 actually holds for *every* DL that enjoys FO-rewritability, including full OWL2 QL. Although interesting from a theoretical perspective, Theorem 3 is not fully satisfactory as it does not tell us *which* CQs are in PTime relative to which TBoxes. In the remainder of this chapter, we carry out a careful inspection of the FO-rewritings obtained in our framework and of the dichotomy result obtained by Dalvi, Suciu and Schnaitter, which results in a more concrete formulation of the dichotomy stated in Theorem 3 and provides a transparent characterization of the PTime cases. For simplicity and without further notice, we concentrate on CQs that are connected, Boolean, and do not contain individual names.

For two CQs q, q' and a TBox $\mathcal{T}$, we say that q *$\mathcal{T}$-implies* q' and write $q \sqsubseteq_{\mathcal{T}} q'$ when $\mathsf{cert}_{\mathcal{T}}(q, \mathcal{A}) \subseteq \mathsf{cert}_{\mathcal{T}}(q', \mathcal{A})$ for all ABoxes $\mathcal{A}$. We say that q and q' are *$\mathcal{T}$-equivalent* and write $q \equiv_{\mathcal{T}} q'$ if $q \sqsubseteq_{\mathcal{T}} q'$ and $q' \sqsubseteq_{\mathcal{T}} q$. We say that q is *$\mathcal{T}$-minimal* if there is no $q' \subsetneq q$ such that $q \equiv_{\mathcal{T}} q'$. When $\mathcal{T}$ is empty, we simply drop it from the introduced notation, writing for example $q \sqsubseteq q'$ and speaking of *minimality*. To have more control over the effect of the TBox, we will generally work with CQs q and TBoxes $\mathcal{T}$ such that q is *$\mathcal{T}$-minimal*. This is without loss of generality because for every CQ q and TBox $\mathcal{T}$, we can find a CQ q' that is $\mathcal{T}$-minimal and such that $q \equiv_{\mathcal{T}} q'$ [4]; note that the answer probabilities relative to $\mathcal{T}$ are identical for q and q'.

We now introduce a class of queries that will play a crucial role in our analysis.

Definition 5 (Simple Tree Queries). *A CQ q is a* simple tree *if there is a variable $x_r \in \mathsf{var}(q)$ that occurs in every atom in q, i.e., all atoms in q are of the form $A(x_r)$, $r(x_r, y)$, or $r(y, x_r)$ ($y = x_r$ is possible). Such a variable x_r is called a* root variable.

As examples, consider the CQs in Figure 1, which are all simple tree queries. The following result shows why simple tree queries are important. A UCQ $\widehat{q}$ is *reduced* if for all disjuncts q, q' of $\widehat{q}$, $q \sqsubseteq q'$ implies $q = q'$.

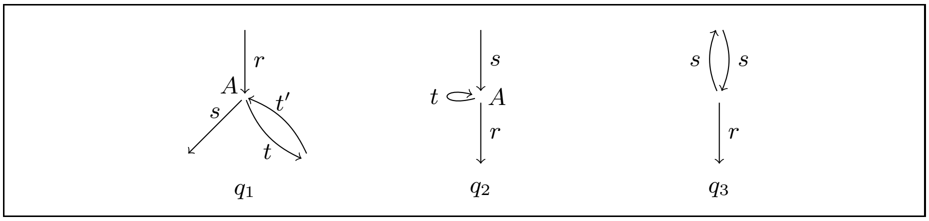


Fig. 1. Example queries

Theorem 4. *Let q be a CQ and $\mathcal{T}$ a DL-Lite TBox such that q is $\mathcal{T}$-minimal and not a simple tree query. Then q is* #P*-hard relative to $\mathcal{T}$*

Proof. (sketch) Let $q_\mathcal{T}$ be a UCQ that is an FO-rewriting of q relative to $\mathcal{T}$. By definition of FO-rewritings, we can w.l.o.g. assume that q occurs as a disjunct of $q_\mathcal{T}$. The following is shown in [8]:

1. if a minimal CQ does not contain a variable that occurs in all atoms, then it is #P-hard over tuple independent databases;
2. if a reduced UCQ $\widehat{q}$ contains a CQ that is #P-hard over tuple independent databases, then $\widehat{q}$ is also hard over tuple independent databases.

Note that since q is $\mathcal{T}$-minimal, it is also minimal. By Points 1 and 2 above, it thus suffices to show that $q_\mathcal{T}$ can be converted into an equivalent *reduced* UCQ such that q is still a disjunct, which amounts to proving that there is no disjunct q' in $q_\mathcal{T}$ such that $q \sqsubseteq q'$ and $q' \not\sqsubseteq q$. The details of the proof, which is surprisingly subtle, are given in the appendix. ❑

To obtain a dichotomy, it thus remains to analyze simple tree queries. We say that a role R *can be generated in a CQ* q if one of the following holds: (i) there is an atom $R(x_r, y) \in q$ and $y \neq x_r$; (ii) there is an atom $A(x_r) \in q$ and $\mathcal{T} \models \exists R \sqsubseteq A$; (iii) there is an atom $S(x, y) \in q$ with x a root variable and such that $y \neq x$ occurs only in this atom, and $\mathcal{T} \models \exists R \sqsubseteq \exists S$. The concrete version of our dichotomy result is as follows. Its proof is based on a careful analysis of FO-rewritings and the results in (the submitted journal version of) [8].

Theorem 5 (Concrete Dichotomy). *Let $\mathcal{T}$ be a DL-Lite TBox. A $\mathcal{T}$-minimal CQ q is in* PTime *relative to $\mathcal{T}$ iff*

1. *q is a simple tree query, and*
2. *if r and r^- are $\mathcal{T}$-generated in q, then $\{r(x, y)\} \sqsubseteq_\mathcal{T} q$ or q is of the form $\{S_1(x, y), \ldots, S_k(x, y)\}$.*

Otherwise, q is #P*-hard relative to $\mathcal{T}$.*

As examples, consider again the queries q_1, q_2, and q_3 in Figure 1 and let $\mathcal{T}_\emptyset$ be the empty TBox. All CQs are $\mathcal{T}_\emptyset$-minimal, q_1 and q_2 are in PTime, and q_3 is #P-hard (all relative to $\mathcal{T}_\emptyset$). Now consider the TBox $\mathcal{T} = \{\exists s \sqsubseteq \exists r\}$. Then q_1 is $\mathcal{T}$-minimal and still in PTime; q_2 is $\mathcal{T}$-minimal, and is now #P-hard because both s and s^- is $\mathcal{T}$-generated. The CQ q_3 can be made $\mathcal{T}$-minimal by dropping the r-atom, and is in PTIME relative to $\mathcal{T}$.

Theorems 4 and 5 show that only very simple CQs can be answered in PTIME. This issue is taken up again in Section 6. We refrain from analyzing in more detail the case where also answer variables and individual names can occur in CQs, and where CQs need not to be connected. It can however be shown that, whenever a connected Boolean CQ q is in PTIME relative to a DL-Lite TBox $\mathcal{T}$, then any CQ obtained from q by replacing quantified variables with answer variables and individual names is still in PTIME relative to $\mathcal{T}$.

5 Beyond Query Rewriting

We have established FO-rewritability as a tool for proving PTIME results for CQ answering in the context of probabilistic OBDA. The aim of this section is to establish that, in a sense, the tool is *complete*: we prove that whenever a CQ q is not FO-rewritable relative to a TBox $\mathcal{T}$, then q is #P-hard relative to $\mathcal{T}$; thus, when a query is in PTIME relative to a TBox $\mathcal{T}$, then this can *always* be shown via FO-rewritability. To achieve this goal, we select a DL as the TBox language that, unlike DL-Lite, also embraces non FO-rewritable CQs/TBoxes. Here we choose $\mathcal{ELI}$, which is closely related to the OWL2 EL profile and properly generalizes DL-Lite (as in the previous sections, we do not explicitly consider the $\bot$ constructor). Note that, in traditional OBDA, there is a drastic difference in data complexity of CQ-answering between DL-Lite and $\mathcal{ELI}$: the former is in AC_0 while the latter is PTIME-complete.

We focus on Boolean CQs q that are *rooted*, i.e., q involves at least one individual name and is connected. This is a natural case since, for any non-Boolean connected CQ $q(\boldsymbol{x})$ and potential answer $\boldsymbol{a}$, the probability $p_{\mathcal{A},\mathcal{T}}(\boldsymbol{a} \in q(\boldsymbol{x}))$ that $\boldsymbol{a}$ is a certain answer to q w.r.t. $\mathcal{A}$ and $\mathcal{T}$ is identical to the probability $p(\mathcal{A}, \mathcal{T} \models q[\boldsymbol{a}])$ that $\mathcal{A}$ and $\mathcal{T}$ entail the rooted Boolean CQ $q[\boldsymbol{a}]$. Our main theorem is as follows.

Theorem 6. *If a Boolean rooted CQ q is not FO-rewritable relative to an $\mathcal{ELI}$-TBox $\mathcal{T}$, then q is #P-hard relative to $\mathcal{T}$.*

Since the proof of Theorem 6 involves some parts that are rather technical, we defer full details to the appendix and present only a sketch of the ideas. A central step is the following observation, whose somewhat laborious proof consists of a sequence of ABox transformations. It uses a notion of boundedness similar to the one introduced in [26], but adapted from instance queries to CQs.

Lemma 1. *If a Boolean rooted CQ q is not FO-rewritable relative to an $\mathcal{ELI}$-TBox $\mathcal{T}$, then there exists an ABox $\mathcal{A}$ and assertions $R_3(a_3, a_2)$, $R_2(a_2, a_1)$, $R_1(a_1, a_0)$ such that $\mathcal{A}, \mathcal{T} \models q$, but $\mathcal{A}', \mathcal{T} \not\models q$ when $\mathcal{A}'$ is $\mathcal{A}$ with any of the assertions $R_3(a_3, a_2), R_2(a_2, a_1), R_1(a_1, a_0)$ dropped.*

We now prove Theorem 6 by a reduction of the problem of counting the number of satisfying assignments for a monotone bipartite DNF formula, which is known to

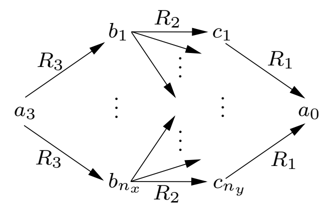


Fig. 2. Gadget for the #P-hardness proof

be #P-hard. The reduction is similar to what was done in [9]. More specifically, input formulas are of the form $\psi = (x_{i_1} \wedge y_{j_1}) \vee \cdots \vee (x_{i_k} \wedge y_{j_k})$ where the set X of variables that occur on the left-hand side of a conjunction in ψ is disjoint from the set Y of variables that occur on the right-hand side of a conjunction in ψ.

For the reduction, let ψ be a formula as above, $X = \{x_1, \dots, x_{n_x}\}$, and $Y = \{y_1, \dots, y_{n_y}\}$. We define an ipABox $(\mathcal{A}_\psi, p_\psi)$ by starting with the ABox $\mathcal{A}$ from Lemma 1 and duplicating the assertions $R_3(a_3, a_2)$, $R_2(a_2, a_1)$, $R_1(a_1, a_0)$ using fresh individual names $b_1, \dots, b_{n_x}$ and $c_1, \dots, c_{n_y}$. This is indicated in Figure 2 where, in the middle part, there is an R_2-edge from every b_i to every c_j. Apart from what is shown in the figure, each b_i receives exactly the same role assertions and outgoing edges that a_2 has in $\mathcal{A}$, and each c_i is, in the same sense, a duplicate of a_1 in $\mathcal{A}$.

In the resulting ipABox $\mathcal{A}_\psi$, every assertion except those of the form $R_3(a_3, b_i)$ and $R_1(c_i, a_0)$ has probability 1; specifically, these are all assertions in $\mathcal{A}_\psi$ that are not displayed in the snapshot shown in Figure 2 and all R_2-edges in that figure. The edges of the form $R_3(a_3, b_i)$ and $R_1(c_i, a_0)$ have probability 0.5. For computing the answer probability $p(\mathcal{A}_\psi, \mathcal{T} \models q)$, one has to consider the ABoxes $\mathcal{A}' \subseteq \mathcal{A}_\psi$ with $p(\mathcal{A}') > 0$. Each such ABox has probability $\frac{1}{2^{|X|+|Y|}}$ and corresponds to a truth assignment $\delta_{\mathcal{A}'}$ to the variables in $X \cup Y$: for $x_i \in X$, $\delta_{\mathcal{A}'}(x_i) = 1$ iff $R_3(a_3, b_i) \in \mathcal{A}'$ and for $y_i \in Y$, $\delta_{\mathcal{A}'}(y_i) = 1$ iff $R_1(c_i, a_0) \in \mathcal{A}'$. Let $\#\psi$ the number of truth assignments to the variables $X \cup Y$ that satisfy ψ. To complete the reduction, we show that $p(\mathcal{A}_\psi, \mathcal{T} \models q) = \frac{\#\psi}{2^{|X|+|Y|}}$. By what was said above, this is an immediate consequence of the following lemma, proved in the appendix.

Lemma 2. *For all ABoxes $\mathcal{A}' \subseteq \mathcal{A}_\psi$ with $p_\psi(\mathcal{A}') > 0$, $\delta_{\mathcal{A}'} \models \psi$ iff $\mathcal{A}', \mathcal{T} \models q$.*

This finishes the proof of Theorem 6. As a by-product, we obtain the following; the proof can be found in the long version.

Theorem 7 ($\mathcal{ELI}$ dichotomy). *Let q be a connected Boolean CQ and $\mathcal{T}$ an $\mathcal{ELI}$-TBox. Then q is in* PTIME *relative to $\mathcal{T}$ or* #P*-hard relative to $\mathcal{T}$.*

6 Monte Carlo Approximation

The results in Sections 4 and 5 show that PTIME complexity is an elusive property even for ipABoxes and relatively inexpressive TBox languages such as DL-

Lite and $\mathcal{ELI}$. Of course, the same is true for probabilistic databases, even for very simple data models such as tuple independent databases. To address this fundamental problem, researchers are often trading accuracy for efficiency, replacing exact answers with approximate ones. In particular, it is popular to use Monte Carlo approximation in the incarnation of a *fully polynomial randomized approximation scheme (FPRAS)*. In this section, we discuss FPRASes in the context of probabilistic OBDA.

An *FPRAS for a Boolean CQ q and TBox $\mathcal{T}$* is a randomized polytime algorithm that, given an ipABox $\mathcal{A}$ and an error bound $\epsilon > 0$, computes a real number x such that

$$\mathsf{Pr}\Big(\frac{|p(\mathcal{A},\mathcal{T} \models q) - x|}{p(\mathcal{A},\mathcal{T} \models q)} \leq \frac{1}{\epsilon}\Big) \geq \frac{3}{4}.$$

In words: with a high probability (the value of $\frac{3}{4}$ can be amplified by standard methods), the algorithm computes a result that deviates from the actual result by at most the factor $\frac{1}{\epsilon}$.

It follows from the proof of Theorem 2 and the fact that there is no FPRAS for the number of satisfying assignments of a propositional formula (unless the complexity classes RP and NP coincide, which is commonly assumed not to be the case) that, over pABoxes, there is no FPRAS for any CQ q and TBox $\mathcal{T}$. Thus, we again have to restrict ourselves to ipABoxes. As observed in [9], it is an easy consequence of a result of Karp and Luby [21] that there is an FPRAS for every CQ over tuple independent databases. By Theorem 1, there is thus also an FPRAS for every CQ q and DL-Lite TBox $\mathcal{T}$ over ipABoxes. The same is true for every FO-rewritable TBox formulated in $\mathcal{ELI}$ or any other TBox language. This observation clearly gives hope for the practical feasibility of probabilistic OBDA.

It is a natural question whether FPRASes also exist for (CQs and) TBoxes formulated in richer ontology languages. No general positive result can be expected for expressive DLs that involve all Boolean operators; the basic such DL is $\mathcal{ALC}$ with concept constructors $\neg C$, $C \sqcap D$, and $\exists r.C$, a typically well-behaved fragment of OWL DL. As analyzed in detail in [26], there is a large class of Boolean CQs q and $\mathcal{ALC}$-TBoxes $\mathcal{T}$ such that, given a non-probabilistic ABox $\mathcal{A}$, it is coNP-hard to check the entailment $\mathcal{A},\mathcal{T} \models q$. A computation problem whose decision version is coNP-hard cannot have an FPRAS [19], and thus we obtain the following.

Theorem 8. *There are CQs q and $\mathcal{ALC}$-TBoxes $\mathcal{T}$ such that there is no FPRAS for q and $\mathcal{T}$.*

In $\mathcal{ELI}$, entailment by non-probabilistic ABoxes can be checked in PTIME for all CQs q and TBoxes $\mathcal{T}$. By what was said above, the interesting cases are those that involve a TBox which is not FO-rewritable. For example, answering the query $A(a)$ and TBox $\{\exists r.A \sqsubseteq A\}$ over ipABoxes roughly corresponds to a directed, two-terminal version of network reliability problems, for which FPRASes can be rather hard to find, see for example [20,34]. We leave a detailed analysis of FPRASes for (CQs q and) $\mathcal{ELI}$-TBoxes $\mathcal{T}$ as interesting future work. Ideally, one would like to have a full classification of all pairs $(q, \mathcal{T})$ according to whether or not an FPRAS exists.

7 Conclusion

We have introduced a framework for ontology-based access to probabilistic data that can be implemented using existing probabilistic database system, and we have analyzed the data complexity of computing answer probabilities in this framework. There are various opportunities for future work. For example, it would be interesting to extend the *concrete* dichotomy from the basic DL-Lite dialect studied in this paper to more expressive versions of DL-Lite that, for example, allow role hierarchy statements in the TBox. It would also be worthwhile to add probabilities to the TBox instead of admitting them only in the ABox; this is done for example in [27,12], but it remains to be seen whether the semantics used there is appropriate for our purposes. Finally, it would be interesting to study the existence of FPRASes for approximating answer probabilities when TBoxes are formulated in $\mathcal{ELI}$.

Acknowledgement. This work was supported by the DFG project Prob-DL (LU1417/1-1).

References

1. Antova, L., Jansen, T., Koch, C., Olteanu, D.: Fast and simple relational processing of uncertain data. In: Proc. of ICDE, pp. 983–992 (2008)
2. Antova, L., Koch, C., Olteanu, D.: 10^{10^6} worlds and beyond: efficient representation and processing of incomplete information. VLDB J. 18(5), 1021–1040 (2009)
3. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F. (eds.): The Description Logic Handbook. Cambridge University Press (2003)
4. Bienvenu, M., Lutz, C., Wolter, F.: Query containment in description logics reconsidered. In: Proc. of KR (2012)
5. Boulos, J., Dalvi, N.N., Mandhani, B., Mathur, S., Ré, C., Suciu, D.: MYSTIQ: a system for finding more answers by using probabilities. In: Proc. of SIGMOD, pp. 891–893 (2005)
6. Calvanese, D., Giacomo, G.D., Lembo, D., Lenzerini, M., Rosati, R.: Tractable reasoning and efficient query answering in description logics: The *DL-Lite* family. J. Autom. Reasoning 39(3), 385–429 (2007)
7. Dalvi, N.N., Ré, C., Suciu, D.: Probabilistic databases: diamonds in the dirt. Commun. ACM 52(7), 86–94 (2009)
8. Dalvi, N.N., Schnaitter, K., Suciu, D.: Computing query probability with incidence algebras. In: Proc. of PODS, pp. 203–214. ACM (2010)
9. Dalvi, N.N., Suciu, D.: Efficient query evaluation on probabilistic databases. VLDB J. 16(4), 523–544 (2007)
10. Dalvi, N.N, Suciu, D.: The Dichotomy of Probabilistic Inference for Unions of Conjunctive Queries. Submitted to Journal of the ACM

11. Finger, M., Wassermann, R., Cozman, F.G.: Satisfiability in $\mathcal{EL}$ with sets of probabilistic ABoxes. In: Proc. of DL. CEUR-WS, vol. 745 (2011)
12. Fuhr, N., Rölleke, T.: A probabilistic relational algebra for the integration of information retrieval and database systems. ACM Trans. Inf. Syst. 15(1), 32–66 (1997)
13. Furche, T., Gottlob, G., Grasso, G., Gunes, O., Guo, X., Kravchenko, A., Orsi, G., Schallhart, C., Sellers, A.J., Wang, C.: Diadem: domain-centric, intelligent, automated data extraction methodology. In: Proc. of WWW, pp. 267–270. ACM (2012)
14. Gottlob, G., Lukasiewicz, T., Simari, G.I.: Conjunctive Query Answering in Probabilistic Datalog+/– Ontologies. In: Rudolph, S., Gutierrez, C. (eds.) RR 2011. LNCS, vol. 6902, pp. 77–92. Springer, Heidelberg (2011)
15. Green, T.J., Tannen, V.: Models for incomplete and probabilistic information. IEEE Data Engineering Bulletin 29(1), 17–24 (2006)
16. Gupta, R., Sarawagi, S.: Creating probabilistic databases from information extraction models. In: Proc. of VLDB, pp. 965–976. ACM (2006)
17. Halpern, J.Y.: An analysis of first-order logics of probability. Artif. Intell. 46(3), 311–350 (1990)
18. Imielinski, T., Lipski Jr., W.: Incomplete information in relational databases. J. of the ACM 31(4), 761–791 (1984)
19. Jerrum, M., Valiant, L.G., Vazirani, V.V.: Random generation of combinatorial structures from a uniform distribution. Theor. Comput. Sci. 43, 169–188 (1986)
20. Karger, D.R.: A randomized fully polynomial time approximation scheme for the all-terminal network reliability problem. SIAM J. Comput. 29(2), 492–514 (1999)
21. Karp, R.M., Luby, M.: Monte-carlo algorithms for enumeration and reliability problems. In: Proc. of FoCS, pp. 56–64. IEEE Computer Society (1983)
22. Kontchakov, R., Lutz, C., Toman, D., Wolter, F., Zakharyaschev, M.: The combined approach to query answering in DL-Lite. In: Proc. of KR. AAAI Press (2010)
23. Laender, A.H.F., Ribeiro-Neto, B.A., da Silva, A.S., Teixeira, J.S.: A brief survey of web data extraction tools. SIGMOD Record 31(2), 84–93 (2002)
24. Lukasiewicz, T., Straccia, U.: Managing uncertainty and vagueness in description logics for the semantic web. J. Web Sem. 6(4), 291–308 (2008)
25. Lutz, C., Schröder, L.: Probabilistic description logics for subjective uncertainty. In Proc. of KR. AAAI Press (2010)
26. Lutz, C., Wolter, F.: Non-uniform data complexity of query answering in description logics. In: Proc. of KR. AAAI Press (2012)
27. Raedt, L.D., Kimmig, A., Toivonen, H.: Problog: a probabilistic prolog and its application in link discovery. In: Proc. of IJCAI, pp. 2468–2473. AAAI Press (2007)
28. Rossmann, B.: Homomorphism preservation theorems. J. ACM 55(3), 1–54 (2008)
29. Sarma, A.D., Benjelloun, O., Halevy, A.Y., Widom, J.: Working models for uncertain data. In: Proc. of ICDE. IEEE Computer Society (2006)
30. Straccia, U.: Top-k retrieval for ontology mediated access to relational databases. Information Sciences 108, 1–23 (2012)
31. Suciu, D., Olteanu, D., Ré, C., Koch, C.: Probabilistic Databases. Synthesis Lectures on Data Management. Morgan & Claypool Publishers (2011)
32. Valiant, L.G.: The complexity of enumeration and reliability problems. SIAM J. Comput. 8(3), 410–421 (1979)
33. Widom, J.: Trio: A system for integrated management of data, accuracy, and lineage. In: Proc. of CIDR, pp. 262–276 (2005)
34. Zenklusen, R., Laumanns, M.: High-confidence estimation of small s-t reliabilities in directed acyclic networks. Networks 57(4), 376–388 (2011)

Predicting Reasoning Performance Using Ontology Metrics

Yong-Bin Kang[1], Yuan-Fang Li[1], and Shonali Krishnaswamy[1,2]

[1] Faculty of IT, Monash University, Australia
[2] Institute for Infocomm Research, A*STAR, Singapore
{yongbin.kang,yuanfang.li,shonali.krishnaswamy}@monash.edu

Abstract. A key issue in semantic reasoning is the computational complexity of inference tasks on expressive ontology languages such as OWL DL and OWL 2 DL. Theoretical works have established worst-case complexity results for reasoning tasks for these languages. However, hardness of reasoning about individual ontologies has not been adequately characterised. In this paper, we conduct a systematic study to tackle this problem using machine learning techniques, covering over 350 real-world ontologies and four state-of-the-art, widely-used OWL 2 reasoners. Our main contributions are two-fold. Firstly, we learn various classifiers that accurately predict classification time for an ontology based on its metric values. Secondly, we identify a number of metrics that can be used to effectively predict reasoning performance. Our prediction models have been shown to be highly effective, achieving an accuracy of over 80%.

1 Introduction

Ontologies are essential building blocks of the Semantic Web. However, the high expressivity of ontology languages also incurs high computational complexity. For example, it has been shown that $\mathcal{SHOIN}(\mathbf{D})$, the description logic (DL) underlying OWL DL, is of worst-case NExpTime-complete complexity [10]. The complexity of $\mathcal{SROIQ}(\mathbf{D})$, the DL underlying OWL 2 DL, is even higher (2NexpTime-complete) [7].

The past decade has seen the development of highly optimized inference algorithms for description logics, with tableau algorithms [2] being a leading exemplar. A number of high-performance tableaux-based reasoners have been developed, including FaCT++ [19], HermiT [15], Pellet [16] and TrOWL [18]. Despite the tremendous progress in ontology reasoning, the high theoretical worst-case complexity results for OWL DL and OWL 2 DL still implies that core reasoning services may be computationally very expensive. For example, it is shown in [4] that although the simpler OWL 2 EL profile has polynomial-time inference algorithms [1], reasoning about large ontologies in OWL 2 EL (Gene Ontology, NCI Thesaurus and SNOMED CT) still requires considerable amounts of time and computational resources.

Moreover, worst-case complexity does not necessarily indicate real-world performance on individual ontologies. In this context, it is noteworthy that reasoner benchmarking has been conducted previously [12,6,4]. However, these works only compared inference performance on a small set of ontologies. They did not attempt to correlate characteristics of ontologies with their inference performance. Hence, they do not provide insight into what makes inference difficult on a given ontology.

Metrics have been used widely and successfully to model artefact complexity in combinatorics and software engineering. We believe that they can be similarly applied to the problem of modelling of ontology inference performance, using a set of ontology metrics like those defined in [24] as a basis. In this paper, we tackle the challenge of predicting ontology classification performance by applying machine learning techniques.

Specifically, we conduct a comprehensive and rigorous investigation, using more than 350 real-world ontologies and 4 widely-used OWL 2 DL reasoners (FaCT++, HermiT, Pellet and TrOWL).[1] Multiple classifiers and feature selection algorithms are tested for their effectiveness. Moreover, 27 metrics are studied for their suitableness for performance prediction. To the best of our knowledge, to date this is the most comprehensive study on characterizing ontology inference performance, and it is the first study on predicting ontology inference performance.

The main contributions of this paper can be summarized as follows:

Prediction Model. We learn a random forest-based classifier that is consistently accurate in predicting ontology classification time using metrics. The accuracy of the classifier is over 90% for HermiT and TrOWL, and over 80% for FaCT++ and Pellet.

Key Metrics. A set of 8 ontology metrics are identified according to their effectiveness in predicting classification performance. These metrics can provide additional insights into ontology engineering and maintenance.

2 Background and Related Work

Works mostly closely related to ours are the ones that compare the performance of OWL reasoners. Benchmarking of description logics (hence ontology) reasoners is not a new topic. However, with the rapid advances made in reasoning algorithms and reasoners, there is sustained interest and need for repeated and rigorous evaluation. Early works [8,9] mainly used synthetic TBoxes for evaluating system performance on less expressive languages ($\mathcal{ALC}$ and its predecessors). In [17], Tempich and Volz developed a systematic method of generating synthetic ontologies. They also performed k-means clustering on 95 medium-sized ontologies (65 class expressions and 25 properties on average) and obtained 3 clusters of ontologies, using a number of different language constructs as features. Wang and Parsia [20] developed Tweezers, a profiler for Pellet, that is capable of

[1] Note that TrOWL is a reasoning infrastructure that is capable of performing incomplete reasoning for OWL 2 DL through approximation [13]. The degree of completeness is not the focus of this paper and hence is not tested.

collecting inference results and performance statistics. The authors demonstrated how such data can be used to modify an ontolgy to speed up reasoning.

In [3], 4 ontologies, each from a language with a different expressivity (RDFS(DL), OWL Lite, OWL DLP and OWL DL) were chosen to compare a number of OWL reasoners. Both TBox and ABox reasoning tasks were compared for a number of reasoners and reason-able triple stores such as Sesame. Reasoner benchmarking has been done using either synthetic or real-world ontologies [12,6]. More recently, 8 modern reasoners are compared on 3 large OWL 2 EL ontologies (Gene Ontology, NCI Thesaurus and SNOMED CT) [4]. Various dimensions of the OWL reasoners were discussed with a focus on performance. The authors drew the conclusion that there is significant performance variability among reasoners and it should be further investigated. This work partially motivated our investigation in this paper.

In the SEALS project,[2] the Storage and Reasoning Systems Evaluation Campaign 2010 aimed at the evaluation of DL-based reasoners. In the evaluation, the performance of three reasoners: FaCT++, HermiT, and jcel were measured and compared in terms of a suite of standard inference services such as classification, class/ontology satisfiability, and logical entailment. Although the evaluation produces a good performance comparison summary of the different reasoners, it does not seem to identify what impact ontology characteristics have on the performance of these reasoners.

There has been research on the development of a series of metrics for analyzing ontology complexity. For example, some metrics have been proposed [22,23] for analyzing ontology complexity by examining the quantity, ratio, and correlativity of classes and relations in a given ontology. However, the metrics developed in this work focused on characteristics of classes without considering a broader range of ontology characteristics. Also, these metrics were mainly designed to analyze complexity evolution and distribution of ontologies, but not for predicting the reasoning performance of ontologies. The work [5] defined some metrics to analyze structural complexity of a given ontology. However, it focused only on analyzing coupling between classes as a measure of ontology complexity. Thus, it does not provide any evidence of how the metrics can be used in analyzing reasoning performance of ontologies.

In [24] we proposed a suite of metrics with the aim of characterizing different aspects of ontology *design complexity*. These metrics consider a broader range of ontology characteristics, and hence are more suitable for the task of performance prediction. All the metrics can be calculated automatically and efficiently, allowing us to leverage them for predicting reasoning performance.

3 Ontology Metrics

In [24] a total of 8 ontology metrics were defined with the aim of measuring different aspects of the design complexity of OWL ontologies. These metrics are defined on a graph representation of an ontology and are used in this paper as

[2] http://www.seals-project.eu

a set of features for predicting reasoner performance. They can be divided into two categories: *ontology-level metrics* (ONT) and *class-level metrics* (CLS). In addition to these 8 metrics, we have defined some other metrics that measure different aspects of an ontology's size and structural characteristics. The metrics are defined on the *asserted* logical axioms in an ontology and they can be divided into two more categories: *anonymous class expressions* (ACE) and *properties* (PRO). For each ontology, we record the sum of each of the CLS, ACE and PRO metrics. Hence there are 27 distinct metrics in total.

Note that syntactic sugar axioms such as `EquivalenceClasses`, `Disjoint Classes` and `PropertyChain` are transformed into pair-wise axioms with a quadratic increase in the number of axioms.

– ***Ontology-level Metrics*** (ONT). The ONT metrics measure the overall characteristics of a given ontology. Besides the 4 metrics defined previously in [24], including SOV (size of vocabulary), ENR (edge-node ratio), TIP (tree impurity) and EOG (entropy of graph), we define 2 new ONT metrics:
  - CYC (Cyclomatic complexity). CYC is defined as $CYC = \#E - \#N + 2 * cc$, where cc is the number of strongly connected components of the ontology represented as a graph. CYC measures the number of linearly independent paths in the ontology graph.
  - RCH (Expression richness). RCH measures the ratio between the number of anonymous class expressions and the total number of class expressions (including named classes).
– ***Class-level Metrics*** (CLS). Classes are first-class citizens in OWL ontologies, hence we use the 4 CLS metrics defined in [24] to capture characteristics of classes in an ontology. These metrics are NOC (number of children), DIT (depth of inheritance), CID (class in-degree), and COD (class out-degree).
– ***Anonymous Class Expressions*** (ACE). ACE are an important ingredient in building up expressive classes. The ACE metrics records, for each kind of anonymous class expression, the number of occurrences in an ontology. There are altogether 9 metrics: enumeration ($ENUM$), negation (NEG), conjunction ($CONJ$), disjunction ($DISJ$), universal/existential quantification (UF/EF) and min/max/exact cardinality ($MNCAR/MXCAR/CAR$).
– ***Properties*** (PRO). Similarly, property declarations and axioms may impact reasoning performance. The 8 PRO metrics record the number of occurrences of each type of property declaration/axiom: object/datatype property declaration (OBP/DTP), functional (FUN), symmetric (SYM), transitive (TRN), inverse functional ($IFUN$), property equivalence (EQV) and inverse (INV).

Note that although there is no metric specifically for ontology expressivity (EL, QL, etc.), such information is implicitly captured by the ACE and PRO metrics as 0 for a metric indicates the absence of a particular language construct.

4 Investigation Methodology

The principal aims of this paper are two-fold: (1) identifying predictive models that accurately estimate reasoning performance of unknown ontologies, and (2) experimentally discovering significant metrics that influence reasoning performance. The key steps in our investigation can be summarized as follows:

Scoping. There are a number of main reasoning tasks on ontologies, including *classification* and *consistency checking*, which are equivalent to each other [2]. We found that classification takes significantly longer than consistency checking, and that there is a significant discrepancy between consistency checking time reported by the reasoners. Thus, we focus on the more difficult reasoning task, classification, and aim to provide insight into prediction models and key metrics embedded in the models. We perform classification on a number of ontologies using different publicly available reasoners. As stated previously, our analysis is conducted on 4 actively-maintained, open-source and widely-used OWL 2 DL reasoners: FaCT++, HermiT, Pellet and TrOWL.

Data Collection. We collect a number of ontologies with varying characteristics, including the application domain, file size, underlying ontology language, etc. We compute, for each ontology in the collection, (1) its metric values as presented in Section 3, and (2) an average performance time for the reasoning task of *ontology classification* for each of the 4 reasoners.

Furthermore, since our goal is to learn classifiers, the continuous reasoning time values need to be *discretized* in order to assign (i.e. classify) ontologies into separate groups (i.e. class labels) based on their reasoning time.

Feature Selection. We hypothesize that different metrics may have different effects on ontology classification performance. *Feature selection* is a very widely-used techniques in data pre-processing to remove irrelevant features. A number of feature selection algorithms are applied to identify and quantitatively study the ontology metrics that have a strong impact on performance. These algorithms typically fall into two categories. *Feature ranking* algorithms (feature selectors) rank the features by a metric and eliminate all features that do not achieve an adequate threshold. *Subset selection* algorithms search the set of possible features for the optimal subset. In this work, we consider 6 different feature selectors, since we are interested in ranking individual features (metrics) and then finding relevant features based on their ranks. These are the information gain (InfoGain), information gain ratio (GainInfo), support vector machine based weighting (SVM), ReliefF-based (ReliefF), symmetrical uncertainty (Symm), and chi-squared statistic (ChiSquared) feature selectors.

Predictive Model Development. In this work, we develop predictive models using classification techniques (in the machine learning sense) to predict reasoning performance of the classification task (in the ontology reasoning sense). In our evaluation, the categories of ontologies are obtained from discretization of the reasoning time of the ontologies for the task, as stated above. Each ontology is represented as a pair consisting of a subset of metrics and the corresponding category. The subset of metrics is found using the feature selectors described above. Given a dataset consisting of a set of ontologies, we choose the training and test data based on standard *10-fold cross validation*, in which each dataset is divided into 10 subsets. Of the 10 subsets, 1 subset is retained as testing data, and the remaining 9 subsets are used as training data. The validation process is then repeated 10 folds (times).

It is well-known that different classifiers tend to produce different prediction performance. Hence, we employ various classifiers and identify the most effective one to build a predictive model for a given dataset. The effectiveness of each classifier is determined through its *classification accuracy* (simply accuracy), often considered to be the best performance indicator for evaluating classifiers.[3] It measures the proportion of correctly classified ontologies against all ontologies in the testing data.

We implement 9 representative classifiers that are available in Weka [21], with the aim of finding the best predictive models for the four reasoners. These are classified into 5 categories: Bayesian classifiers (BayesNet (BN) and NaïveBayes (NB)), decision tree-based classifiers (J48, RandomForest (RF), REP Tree (RT)), rule-based classifiers (DecisionTable (DT)), a regression-based classifier (SimpleLogistic (SL)), and lazy classifiers (IBk$_{[1 \leq k \leq 10]}$ and K*).

Key Metrics Determination. Identifying the metrics that most highly impact reasoning time can provide insights for ontology engineering. In this step, such metrics are identified by further analyzing outcomes of the feature selectors utilized in the previous step. More specifically, by examining the metrics used in the classifier chosen in the predictive model for each dataset, we can identify which metrics, in conjunction with the classifier, contribute most to accuracy.

Given a dataset of metrics for each reasoner, we apply the 9 classifiers on various subsets of metrics that are identified by the 6 feature selectors. Then, we identify the best predictive model for the reasoner consisting of the following three dimensions: (1) a particular classifier leading to the best accuracy, (2) a particular metric subset, used for the classifier, and a specific feature selector that has found the subset, and (3) the prediction performance (accuracy) result achieved by the classifier with the metric subset. The discovered metric subset for each reasoner is designated as key metrics leading to its best predictive model. Furthermore, we measure the impact of individual metrics with respect to constructing predictive models for the 4 reasoners based on statistical analysis.

5 Data Collection

A total of 358 real-world, public-domain ontologies are collected for this work. No preprocessing (cleansing) is done. A large number of these ontologies are collected from the Tones Ontology Repository and NCBO BioPortal.[4] These ontologies vary in file size, ranging from less than 4KB to almost 300MB. However, it is worth noting that file size is not a very good indicator of reasoning performance, as a small ontology (such as the DOLCE ontology) may `owl:imports` a large number of other ontologies, which make up the potentially very large

[3] F-measure is measured and found to be completely positively correlated to accuracy. For brevity reasons, we only report our experimental results in accuracy.

[4] `http://owl.cs.manchester.ac.uk/repository/`, `http://www.bioontology.org/`

import closure that a reasoner considers. Note that all ontologies collected from BioPortal are large, with at least 10,000 *terms*. The expressivity of these ontologies ranges from OWL 2 EL and QL to OWL Full. At the same time, this collection also includes some well-known hard ontologies such as FMA, DOLCE, Galen, NCI Thesaurus and the Cell Cycle Ontology (CCO).

The values of all metrics are calculated; and the distribution of 8 representative metrics are shown in Figure 1, where the metric values are plotted in log scale and ranked by the values. As can be seen quite clearly, the values for these metrics span a large range, from 0 to more than 10^5, and to more than 10^7 for DIT. Moreover, as expected, the majority of ontologies have metric values in the middle of the range, with a few having values closer to the boundary.

Classification time for all ontologies is also collected. All the experiments are performed on a high-performance server running OS Linux 2.6.18 and Java 1.6 on an Intel (R) Xeon X7560 CPU at 2.27GHz with a maximum of 40GB allocated to the 4 reasoner.[5] OWLAPI version 3.2.4 is used to load ontologies and interface with the reasoners. The reasoners that are invoked are: FaCT++ 1.5.3, HermiT 1.3.5, Pellet 2.3.0 and TrOWL 0.8. REL is the underlying reasoner used by TrOWL. These metrics will be revisited in Section 7.

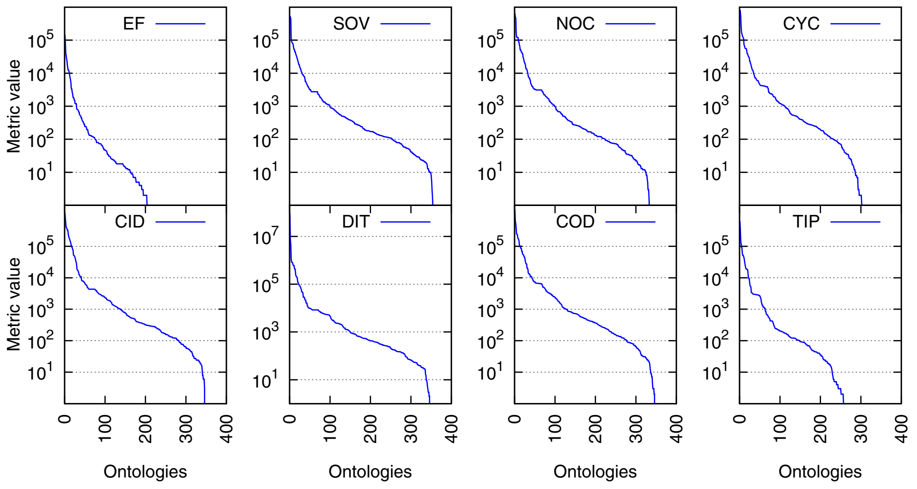


Fig. 1. Distributions of values of 8 metrics

For each ontology and each reasoner, CPU time for classification is averaged over 10 independent runs and recorded. Loading and pairwise subsumption test time is are not included. Trivially simple ontologies (with reasoning time ≤ 0.01s) are later excluded from the experiment to reduce the skewness of the dataset. Some hard ontologies take an extremely long time to classify. Hence, we apply a

[5] To accommodate large ontologies and potential memory leaks in reasoners (due to repeated invocations).

50,000-second cutoff for all the reasoners. The distribution of the raw reasoning time for the four reasoners can be found in Figure 2, where classification time (in log scale) is ordered and plotted against the ontologies. It can be observed that FaCT++, HermiT and Pellet all have some ontologies reaching the cut-off time, while TrOWL successfully classifies all ontologies.[6] It can also be seen that for relatively easy ontologies ($\leq$ 10s), FaCT++ and TrOWL seem to dominate the other 2 reasoners. Compared to performance data reported in [4], the performance on the same ontologies (GO and NCI Thesaurus) seems to be much worse in our experiments, running the same reasoners. Upon closer inspection we notice that the versions of the "same" ontologies are different – we are using more recent versions (current as of November 2011) of these ontologies, which are much larger than those versions used in [4].

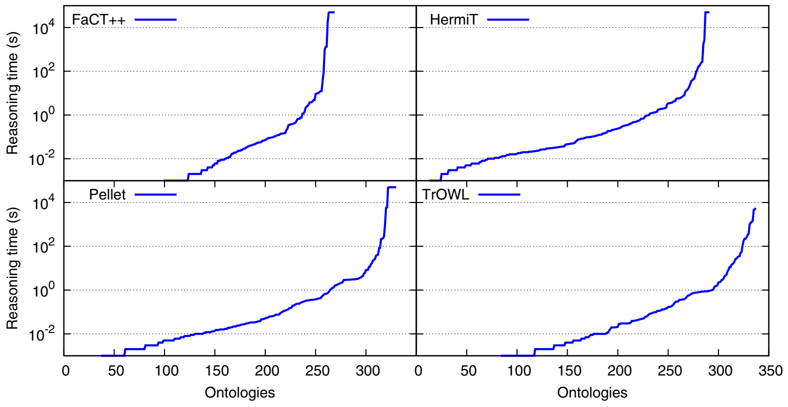


Fig. 2. Raw classification time of the four reasoners

As stated in the previous section, discretization is a necessary first step before classifiers can be trained. After raw run time values are collected, they are discretized into 5 bins, where the bin 'T' contains the trivially simple ontologies (classification time $\leq$ 0.01s). The other 4 bins are of unit interval width. The interval width is used as the exponent of the reasoning time, i.e., 10^i is the cutoff point between bin i and bin $i + 1$, $1 \leq i \leq 4$. These 4 bins are labelled 'A', 'B', 'C' and 'D'. A summary of the discretization and the number of ontologies for each reasoner in each bin is shown in Table 1. It can be seen in the table that each reasoner fails to perform classification on a number of ontologies due to parsing or processing errors or the ontology being inconsistent.

It is worth pointing out that the server where the experiments are performed is very capable. Although 100 seconds is not a very long time, the same ontology will take much longer to run on a less powerful computer (mobile devices in particular).

[6] We note again that this may be due to TrOWL's incomplete reasoning approach.

Table 1. Discretization of reasoning time and number of ontologies in each bin

Discretized label	Classification time	Fact++	HermiT	Pellet	TrOWL
T	T $\leq$ 0.01s	161	77	138	188
A	0.01s $<$ A $\leq$ 1s	75	154	126	105
B	1s $<$ B $\leq$10s	16	35	38	17
C	10s $<$ C $\leq$ 100s	6	12	12	13
D	100s $<$ D	11	13	16	14
Total discretized		269	291	330	337
Ontologies in error		89	67	28	21

More analysis of the performance characteristics of the reasoners can be found in [11]. All the ontologies, their metric values and reasoning time can be found at `http://www.csse.monash.edu/~yli/metrics_perf/`.

6 Predictive Models

In this section, we present the first contribution of our work, the construction and analysis of predictive models for classification performance. Our analysis shows that consistently high accuracy ($>$ 80%) is achieved for all of the 4 reasoners.

Using 9 classifiers and 6 feature selectors, we learn predictive models as specified in Section 4. For each classifier, the 6 feature selectors are applied to find the best set of metrics. The set of metrics leading to the best accuracy for the classifier and the feature selector is then recorded. The accuracy values of the 9 classifiers are measured. More specifically, an accuracy value is measured for each classifier with 6 different collections of best metrics identified by each of the 6 feature selectors. Eventually, a single set of the best metrics for each (classifier, feature selector) pair is collected.

Section 6.1 presents and analyzes the overall accuracy results of the 4 reasoners. Section 6.2 further characterizes the best predictive model and discusses the effect of feature selection.

6.1 Accuracy Distribution and Analysis

For the 4 reasoners, the accuracy distributions of the 9 classifiers (across the 6 feature selectors) are measured and presented in boxplots in Figure 3. Boxplots provide an excellent visual summary of a distribution through 5 statistical measures: *minimum data value* (MIN), *lower quartile* (Q1), *median* (Q2), *upper quartile* (Q3), *maximum data value* (MAX). Further, we enhance the boxplots by additionally showing the *mean* (AVG) of the accuracy data measured for a single classifier across the 6 feature selectors.

A box itself contains the middle 50% of the accuracy data measured by a classifier with the 6 feature selectors; the upper box area (in blue) denotes the 50th$-$75th percentile (Q2$-$Q3) of the data, and the lower box area (in yellow) denotes the 25th$-$50th percentile (Q1$-$Q2). The remaining 50% of the data is contained within the areas between the box and the vertical lines or “whiskers”.

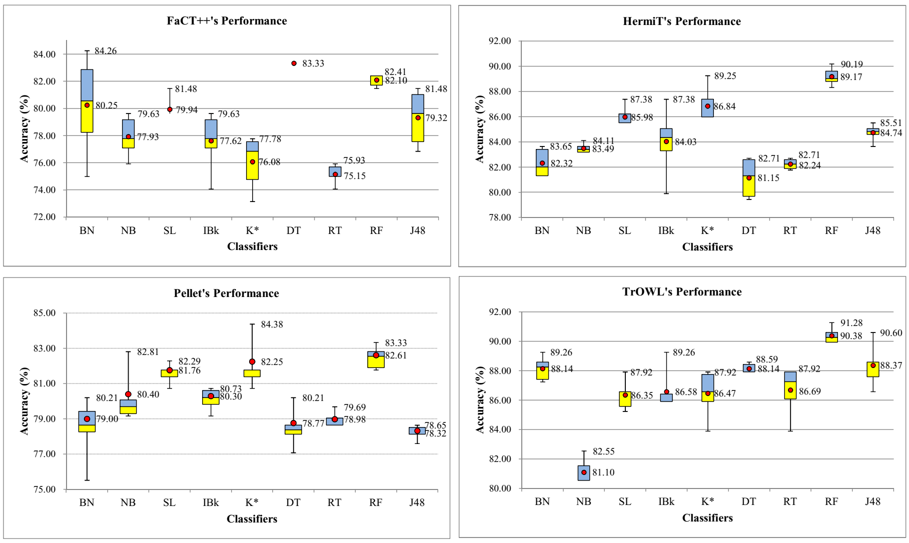


Fig. 3. Boxplots displaying the accuracy distribution for the 4 reasoners

The ends of the whiskers indicate the minimum and maximum accuracy values. The line inside the box indicates the median value of the accuracy data. The mean of the accuracy is represented by a red circle in the box. Among the above 6 statistical measures, the values of 2 measures, maximum (MAX) and mean (AVG) accuracy, are shown in the plot.

A number of important observations can be made from Figure 3.

– *RF (RandomForest) is the most stable predictive model.* For all the 4 reasoners' performance, RF has the smallest difference of 1.42 between MIN and MAX, while BN has the largest of 4.57. This indicates that RF leads to the most reliable and stable accuracy results, while BN leads to the most variable accuracy results.
– *Ontology metrics entail good predictive models.* The range of the MAX accuracy for the 4 reasoners is from 84.26 (by BN for FaCT++) to 91.28 (by RF for TrOWL). This indicates that particular subsets of ontology metrics, identified by different feature selectors, can be effectively leveraged for building good predictive models for classification reasoning performance of the reasoners.
– *RF is the best classifier leading to the best predictive models for the 4 reasoners.* We examine which classifiers lead to the best predictive models for the 4 reasoners through statistical analysis of central tendency (e.g. mean) of the measured quantitative values. We compute the mean of the 6 quantitative values shown in Figure 3 across the 4 reasoners. The results are presented in Table 2. The best result in the mean values for each criterion is denoted in **boldface**. Table 2 clearly shows that RF leads to the best predictive models for all the 4 reasoners for all the 6 measures. In the following section, we describe these models in more detail.

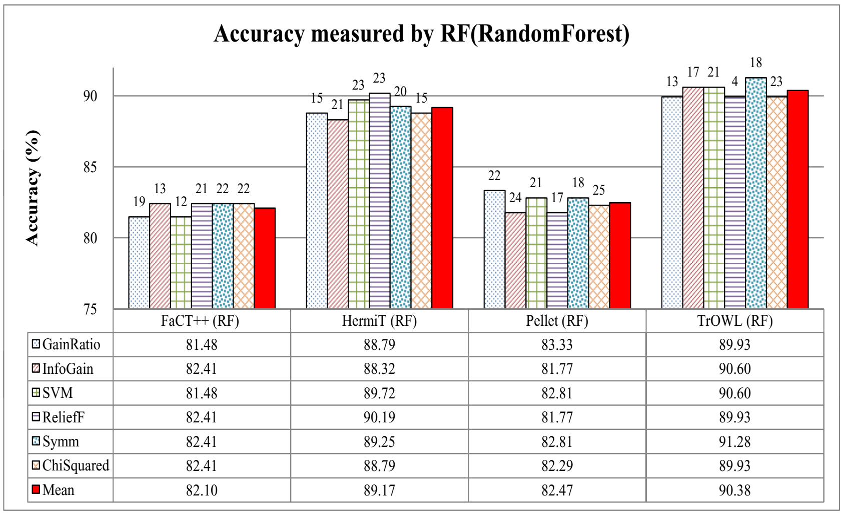
Table 2. The mean quantitative values of the 9 classifiers across the 4 reasoners

Classifier	6 Statistical Measures					
	MIN	Q1	Q2	AVG	Q3	MAX
BN	81.38	82.89	83.41	83.94	84.74	82.30
NB	79.85	79.94	80.23	80.79	82.68	80.66
SL	82.98	83.54	84.12	84.17	85.11	83.47
IBk	82.54	82.88	83.11	83.51	85.00	82.09
K*	82.31	83.70	84.16	84.72	86.15	82.83
DT	82.50	83.04	83.19	83.51	84.40	82.78
RT	81.27	82.40	82.91	83.43	83.80	80.75
RF	**85.85**	**85.97**	**86.41**	**86.68**	**87.30**	**86.03**
J48	82.09	82.90	83.34	83.53	84.63	82.66

6.2 Best Predictive Models

As each reasoner employs a different set of algorithms and optimization techniques, they may exhibit significantly different performance on the same ontology. As a result, the performance of classifiers may be different for the 4 reasoners as well. In this subsection, we further analyze the best classifiers and feature selectors to understand the reasoner-specific behaviours.

As discussed in the previous subsection, RandomForest (RF) is the overall best classifier. This may in part be due to the nature of RF – that it is an *ensemble classifier* that consists of many decision trees. Figure 4 shows, for RF and each reasoner, the MAX classification accuracy (%) for each feature selector, and also their average. The numeric label on top of each bar denotes the number of metrics identified by the corresponding feature selector.



	FaCT++ (RF)	HermiT (RF)	Pellet (RF)	TrOWL (RF)
▫ GainRatio	81.48	88.79	83.33	89.93
▨ InfoGain	82.41	88.32	81.77	90.60
⊞ SVM	81.48	89.72	82.81	90.60
☰ ReliefF	82.41	90.19	81.77	89.93
▩ Symm	82.41	89.25	82.81	91.28
▧ ChiSquared	82.41	88.79	82.29	89.93
🟥 Mean	82.10	89.17	82.47	90.38

Fig. 4. Best predictive models for the 4 reasoners

RF achieves consistently high accuracy, all higher than 80% for each reasoner with an overall average of 86.03%. For TrOWL, RF achieves 90.38% accuracy on average. It suggests that RF can be effectively used in predicting classification reasoning performance. It also reinforces our belief that ontology metrics can be effective in learning predictive models for the reasoning performance. Moreover, it also opens up the potential to apply our approach to predicting reasoning performance of other reasoners.

It can be observed that each best accuracy result comes with a different number of ontology metrics. The numbers vary from 4 (ReliefF for Pellet) to 25 (ChiSquared for FaCT++). Note that not once is the entire set of 27 metrics chosen by any feature selector. This finding establishes the validity of our hypothesis, presented in Section 4, that feature selectors can be leveraged to discover more significant metrics that impact on building more strong predictive models for classification reasoning performance.

7 Key Metrics Identification

In this section, we present the second main contribution of this work, the identification of important metrics that have a strong impact on classification performance. Such knowledge can contribute the task of ontology engineering and maintenance. This identification is achieved through a rigorous quantification of *impact factors* of all 27 ontology metrics used in the classifiers constructed in the previous section.

As discussed in the previous section (Table 2), all of the 9 classifiers achieve mean accuracy of at least 80% for all the 4 reasoners. Such high accuracy makes the case for investigating the metrics used by all the classifiers and feature selectors. Two factors influence the significance of a given metric: (1) how frequently it gets selected to be used in the classifiers, and (2) how much it contributes to prediction of reasoning performance. In other words, the more frequently a metric is used in the predictive models (as chosen by the feature selectors), and the more weight it has in the classifiers that use it, the more it influences ontology classification performance. Hence, we combine these two factors to calculate the *impact factor* of all the metrics.

Let metrics be denoted $m_i, 1 \leq i \leq 27$, classifiers be denoted by $c_j, 1 \leq j \leq 9$, feature selectors be denoted $f_k, 1 \leq k \leq 6$, and reasoners be denoted $r_l, 1 \leq l \leq 4$. We denote with $fs_{j,k,l}$ the set of metrics selected for each classifier c_j by feature selector f_k for reasoner r_l. We further denote with $\#fs_{j,l|i}$ the total number of occurrences of metric m_i in all metric sets identified by the feature selectors for classifier c_j and reasoner r_l $(0 \leq \#fs_{j,l|i} \leq 6)$.

Similarly, let $r_{i,k}^{j,l}$ denote the *weight* of the metric m_i assigned by feature selector f_k for the pair (c_j, r_l) ($r_{i,k}^{j,l} = 0$ if m_i is not selected), normalized by $\max(r_{*,k}^{j,l})$ so that it is between $[0, 1]$. We average over all the feature selectors to obtain the average ranked weight $r_{i,j}^{l} = \frac{\sum_{k=1}^{6} r_{i,k}^{j,l}}{6}$ of m_i.

Algorithm 1 describes the calculation of the *impact factor* for all the metrics.

```
   Input: Metric number of occurrences $f(i,j,k,l)$
   Input: Metric weight $r(i,j,k,l)$
   Output: Impact factor for each metric $\mathbf{mif}_i, 1 \le i \le 27$
 1 foreach reasoner $r_l$ do
 2     Initialize $27 \times 9$ matrices $mft^l, mf^l, r^l, nmf^l$
 3     $mft^l_{i,j} \leftarrow \#fs_{j,l|i}$                  /* Metric frequency per classifier */
 4     $mf^l_{i,j} \leftarrow \frac{mft^l_{i,j}}{\sum_{i=1}^{27} mft^l_{i,j}}$        /* Normalization */
 5     $r^l_{i,j} \leftarrow \frac{\sum_{k=1}^{6} r^{j,l}_{i,k}}{6}$             /* Average ranked weight */
 6     $mf^l \leftarrow mf^l \circ r^l$                 /* Combining frequency with weight */
 7     $nmf^l_{i,j} \leftarrow a + (1-a) \times \frac{mf^l_{i,j}}{\max(mf^l_{*,j})}$  /* Max frequency normalization */
 8 end
 9 Initialize $27 \times 9$ matrix $\mathbf{nmf}$ for each $(m_i, c_j)$, $27 \times 1$ vector $\mathbf{mif}$ for each $m_i$
10 $\mathbf{nmf} \leftarrow \frac{\sum_{l=1}^{4} nmf^l}{4}$              /* Average over the reasoners */
11 $\mathbf{mif}_i \leftarrow \frac{\sum_{j=1}^{9} \mathbf{nmf}_{i,j}}{9}$          /* Average over the classifiers */
12 $\mathbf{mif}_i \leftarrow \frac{\mathbf{mif}_i}{\max(\mathbf{mif})}$           /* Normalization over max */
13 return mif
```

Algorithm 1: The calculation of the impact factor of metrics

For each reasoner (lines 1-8), the combined scores taking into account number of occurrences and weight for each metric are calculated. On line 3, we calculate the total number of occurrences of each metric for each classifier in a 27×9 matrix mft^l (*metric frequency table*). For example, if a metric 'SOV' (denoted m_2) is in the sets of metrics selected by 4 out of the 6 feature selectors for the classifier RF (denoted c_1) and reasoner Pellet (denoted r_3), then $mft^3_{2,1} = 4$. Each $mft^l_{i,j}$ value is then normalized by dividing by the total number of occurrences of all metrics for classifier c_j on line 4.

From the raw weight $r(i,j,k,l)$, we obtain the weight $r^l_{i,j}$ by averaging over all the 6 feature selectors on line 5. Line 6 then combines the frequency and the averaged weight of each metric by taking the entrywise product of matrices mf^l and r^l.

Note that one problem of the measure mf^l as calculated on line 6 is that the difference of impact factors between higher and lower frequency metrics tend to be too large. For example, it seems unlikely that 6 occurrences of a metric with a ranked scored r^l in the collection of a classifier in mft^l truly carry 6 times the significance of a single occurrence with the same or similar ranked score r^l. To avoid this problem, we apply a normalization technique similar to maximum term frequency normalization [14] to mf^l on line 7 to obtain the normalized impact factor values for each (metric, classifier) pair for each reasoner. Parameter a is a value between 0 and 1 and is generally set to 0.4 [14]. It is a *smoothing* term whose role is to dampen the contribution of $\frac{mf^l_{i,j}}{\max(mf^l_{*,j})}$. In this step we also scale down $mf^l_{i,j}$ by the largest frequency values of all metrics in the collection for a classifier c_j.

Eventually, the impact factor values are averaged over all 4 reasoners and all 9 classifiers to obtain the final impact factor values of the metrics (lines 10-11).

The metrics can be grouped into a number of categories according to the quartiles their **mif** values fall into: *Strong Impact* (SI, $0.75 < \textbf{mif} \leq 1$), *Normal Impact* (NI, $0.5 < \textbf{mif} \leq 0.75$), *Weak Impact* (WI, $0.25 < \textbf{mif} \leq 0.5$), and *Very Weak Impact* (VI, $0 \leq \textbf{mif} \leq 0.25$). The ranking and categorization results are shown in Figure 5.

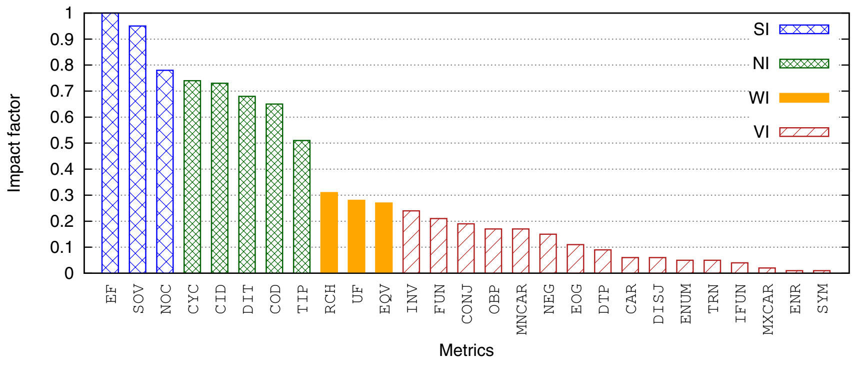


Fig. 5. Normalized impact factors of all the ontology metrics

The 11 metrics in SI, NI and WI are all commonly used in the best predictive models presented in Figure 4. Except in one case, all metric sets selected by feature selectors for RF (in Figure 4) are a superset of the 11 metrics in SI, NI and WI in Figure 5. The exception is the set of metrics selected by ReliefF for TrOWL, where there are only 4 metrics. In this case, however, the 4 metrics also belong to these 3 categories. It can be concluded that these 11 metrics form a core group of metrics that are important for predicting classification performance.

Furthermore, it can be observed in Figure 5 that a group of 8 metrics (SI and NI) have high impact on reasoning performance, and that there is a clear separation of **mif** scores between these two groups of metrics and the rest of the metrics. Specifically, it can be clearly seen that (1) the number of existential quantification restrictions (EF), (2) the size of an ontology (SOV), (3) the number of *independent paths* (CYC), (4) the characteristics of named classes (the 4 CLS metrics) and (5) the non-treelike-ness of the inheritance graph (TIP) have a strong impact on prediction performance.

As mentioned previously, the impact factor indicates a metric's relative influence on (the prediction of) classification performance. The 8 metrics identified above can hence be used to guide ontology engineering. For example, reducing the size of the ontology (SOV), reducing the number of independent paths (CYC), reducing the degree of classes (CID and COD) and making the inheritance graph more tree-like (TIP) may significantly improve reasoning performance of the ontology.

8 Conclusion

Terminological reasoning has been shown to be a computationally expensive problem, especially for expressive languages such as OWL DL and OWL 2 DL. Despite tremendous progress in the past decade in the design and development of highly optimized algorithms and reasoners, ontology classification is still a very challenging task, as demonstrated by previous benchmarking works and our own experiments. It is therefore highly desirable to be able to quantitatively analyze and predict reasoning performance using syntactic features.

Metrics have been successfully used to capture different aspects of the syntactic/structural characteristics of various kinds of artefacts (software, combinatorial problems, etc.), including their complexity and empirical hardness. In this paper we propose, develop and evaluate the use of ontology metrics as an effective basis for predict reasoning time for the task of ontology classification. To the best of our knowledge, this is the first such study to apply machine learning techniques (classification) to predict reasoning time for ontologies.

Our contributions in this paper are three-fold: (1) the development of highly effective (over 80% accuracy) predictive models to estimate the reasoning time for an ontology given its metric values, for four widely-used OWL 2 DL reasoners, (2) the identification of a set of 8 metrics that have the most impact/correlation with reasoning performance, and (3) a rigorous empirical validation of the proposed methodology with a set of over 350 real-world ontologies, the largest study so far in terms of the size of the dataset.

A number of future directions are planned for this work. We will further study the statistical significance of our predictive models and key metrics. Other metrics, such as language profile and number of (hidden) GCIs, will be investigated to evaluate their effectiveness in reasoning time prediction. The effect of optimisation techniques on reasoning performance will also be investigated. We also plan to investigate other reasoning tasks (consistency checking) and other machine learning techniques (regression analysis). The degree of incompleteness of TrOWL will also be studied to quantify its impact on prediction accuracy. Lastly, we will study the feasibility of generating synthetic ontologies with specified reasoning performance with metric values as parameters. Such ontologies will be very valuable in the analysis and optimization of reasoning algorithms.

References

1. Baader, F., Brandt, S., Lutz, C.: Pushing the $\mathcal{EL}$ envelope further. In: Clark, K., Patel-Schneider, P.F. (eds.) Proceedings of the OWLED 2008 DC Workshop on OWL: Experiences and Directions (2008)
2. Baader, F., Sattler, U.: An overview of tableau algorithms for description logics. Studia Logica 69(1), 5–40 (2001)
3. Bock, J., Haase, P., Ji, Q., Volz, R.: Benchmarking OWL reasoners. In: ARea2008 - Workshop on Advancing Reasoning on the Web: Scalability and Commonsense (June 2008)
4. Dentler, K., Cornet, R., ten Teije, A., de Keizer, N.: Comparison of reasoners for large ontologies in the OWL 2 EL profile. Semantic Web Journal 2(2), 71–87 (2011)
5. García, J., García, F., Therón, R.: Defining Coupling Metrics among Classes in an OWL Ontology. In: García-Pedrajas, N., Herrera, F., Fyfe, C., Benítez, J.M., Ali, M. (eds.) IEA/AIE 2010, Part II. LNCS, vol. 6097, pp. 12–17. Springer, Heidelberg (2010)
6. Gardiner, T., Horrocks, I., Tsarkov, D.: Automated benchmarking of description logic reasoners. In: Proceedings of the 2006 International Workshop on Description Logics (DL 2006) (2006)
7. Grau, B.C., Horrocks, I., Motik, B., Parsia, B., Patel-Schneider, P., Sattler, U.: OWL 2: The next step for OWL. Journal of Web Semantics: Science, Services and Agents on the World Wide Web 6, 309–322 (2008)
8. Heinsohn, J., Kudenko, D., Nebel, B., Profitlich, H.-J.: An empirical analysis of terminological representation systems. In: Proceedings of the Tenth National Conference on Artificial Intelligence, AAAI 1992, pp. 767–773. AAAI Press (1992)
9. Horrocks, I., Patel-Schneider, P.F.: DL systems comparison (summary relation). In: Proceedings of the 1998 International Workshop on Description Logics (DL 1998). CEUR Workshop Proceedings, vol. 11. CEUR-WS.org (1998)
10. Horrocks, I., Patel-Schneider, P.F., van Harmelen, F.: From $\mathcal{SHIQ}$ and RDF to OWL: The Making of a Web Ontology Language. Journal of Web Semantics 1(1), 7–26 (2003)
11. Kang, Y.-B., Li, Y.-F., Krishnaswamy, S.: A rigorous characterization of reasoning performance – a tale of four reasoners. In: Proceedings of the 1st International Workshop on OWL Reasoner Evaluation (ORE 2012) (June 2012)
12. Pan, Z.: Benchmarking DL reasoners using realistic ontologies. In: Grau, B.C., Horrocks, I., Parsia, B., Patel-Schneider, P.F. (eds.) OWLED. CEUR Workshop Proceedings, vol. 188. CEUR-WS.org (2005)
13. Ren, Y., Pan, J.Z., Zhao, Y.: Soundness preserving approximation for tbox reasoning. In: Fox, M., Poole, D. (eds.) AAAI. AAAI Press (2010)
14. Salton, G., Buckley, C.: Term-weighting approaches in automatic text retrieval. Information Processing and Management: an International Journal 24(5), 513–523 (1988)
15. Shearer, R., Motik, B., Horrocks, I.: HermiT: A Highly-Efficient OWL Reasoner. In: Proceedings of the 5th International Workshop on OWL: Experiences and Directions, OWLED 2008 (2008)
16. Sirin, E., Parsia, B., Grau, B., Kalyanpur, A., Katz, Y.: Pellet: A practical OWL-DL reasoner. Web Semantics: Science, Services and Agents on the World Wide Web 5(2), 51–53 (2007)
17. Tempich, C., Volz, R.: Towards a benchmark for semantic web reasoners - an analysis of the DAML ontology library. In: Sure, Y., Corcho, Ó. (eds.) EON. CEUR Workshop Proceedings, vol. 87. CEUR-WS.org (2003)

18. Thomas, E., Pan, J.Z., Ren, Y.: TrOWL: Tractable OWL 2 Reasoning Infrastructure. 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. 431–435. Springer, Heidelberg (2010)
19. Tsarkov, D., Horrocks, I.: FaCT++ Description Logic Reasoner: System Description. In: Furbach, U., Shankar, N. (eds.) IJCAR 2006. LNCS (LNAI), vol. 4130, pp. 292–297. Springer, Heidelberg (2006)
20. Wang, T.D., Parsia, B.: Ontology Performance Profiling and Model Examination: First Steps. 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.) ISWC/ASWC 2007. LNCS, vol. 4825, pp. 595–608. Springer, Heidelberg (2007)
21. Witten, I.H., Frank, E.: Data mining: Practical machine learning tools and techniques with Java implementations. Morgan Kaufmann, San Francisco (2000)
22. Yang, Z., Zhang, D., Ye, C.: Evaluation metrics for ontology complexity and evolution analysis. In: IEEE International Conference on E-Business Engineering, pp. 162–170 (2006)
23. Zhang, D., Ye, C., Yang, Z.: An Evaluation Method for Ontology Complexity Analysis in Ontology Evolution. In: Staab, S., Svatek, V. (eds.) EKAW 2006. LNCS (LNAI), vol. 4248, pp. 214–221. Springer, Heidelberg (2006)
24. Zhang, H., Li, Y.-F., Tan, H.B.K.: Measuring Design Complexity of Semantic Web Ontologies. Journal of Systems and Software 83(5), 803–814 (2010)

Formal Verification of Data Provenance Records

Szymon Klarman[1], Stefan Schlobach[1], and Luciano Serafini[2]

[1] VU University Amsterdam, The Netherlands
{s.klarman,k.s.schlobach}@vu.nl
[2] Fondazione Bruno Kessler, Trento, Italy
serafini@fbk.eu

Abstract. Data provenance is the history of derivation of a data artifact from its original sources. As the real-life provenance records can likely cover thousands of data items and derivation steps, one of the pressing challenges becomes development of formal frameworks for their automated verification.

In this paper, we consider data expressed in standard Semantic Web ontology languages, such as OWL, and define a novel verification formalism called *provenance specification logic*, building on dynamic logic. We validate our proposal by modeling the test queries presented in The First Provenance Challenge, and conclude that the logic core of such queries can be successfully captured in our formalism.

1 Introduction

In this paper, we propose and study a novel logic-based approach to formal verification of data provenance records in the Semantic Web environment.

Motivation. Data provenance is the history of derivation of a data artifact from its original sources [1,2]. A provenance record stores all the steps and contextual aspects of the entire derivation process, including the precise sequence of operations executed, their inputs, outputs, parameters, the supplementary data involved, etc., so that third parties can unambiguously interpret the final data product in its proper context. It has been broadly acknowledged that provenance information is crucial for facilitating reuse, management and reproducibility of published data [3,1]. For instance, the ability of verifying whether past experiments conformed to some formal criteria is a key in the process of validation of eScientific results [4]. As provenance records can cover thousands of data items and derivation steps, one of the pressing challenges becomes development of formal frameworks and methods to automate verification. Such a logic back-end for practical reasoning tools could, e.g. be useful for provenance-driven data querying, or for validating conformance of provenance records to formal specifications.

Let us consider a concrete example taken from The First Provenance Challenge (FPC) — a community effort aimed at understanding the capabilities of available provenance systems [5]. In FPC, 17 teams competed in answering 9 queries over data provenance records (see Figure 1) obtained from executing a

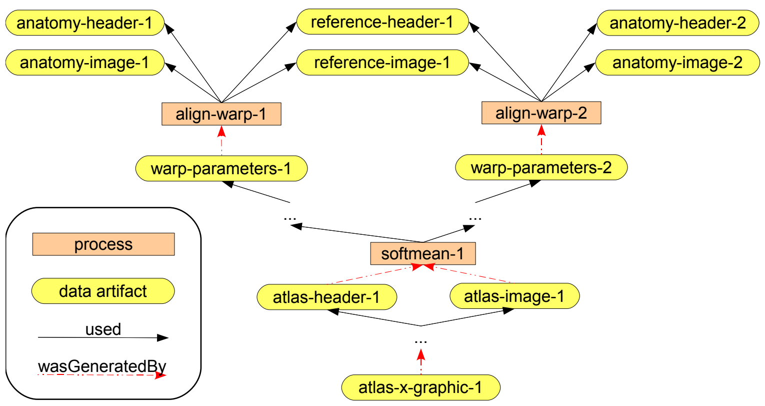


Fig. 1. A data provenance record describing a run of the FPC workflow [5]

real-life scientific workflow for creating population-based "brain atlases" of high resolution anatomical data. One representative task was to:

Q6. *Find all output averaged images of softmean (average) procedures, where the warped images taken as input were align warp'ed using a twelfth order nonlinear 1365 parameter model, i.e. where softmean was preceded in the workflow, directly or indirectly, by an align warp procedure with argument -m 12.*

A distinctive feature of this sort of queries is their inherent two-dimensionality: the domain data (here: image identifiers) is queried relative to its meta-level provenance description. To date all existing approaches to support such queries are based on ad hoc combinations of techniques and formalisms, dependent on the internal representation structures, and are procedural in nature. Given the semantic character of the task, and in light of the soon to be expected standardization of the Provenance vocabularies by W3C,[1] a logic-based language for querying and verifying provenance graphs, which could significantly improve reusability and generalisability, is critically missing. This paper closes this gap.

Methodology. We introduce *provenance specification logic* (PSL^M) which, to the best of our knowledge, offers the first systematic view on the logical foundations of formal verification of data provenance records. Our focus is on data expressed in the Semantic Web ontology languages, such as OWL and RDF(S), whose formal core is essentially captured by Description Logics (DLs) [6], underpinning the Semantic Web architecture.

The basic idea is very intuitive. A data provenance record is represented as a directed graph, with certain nodes being treated as identifiers for datasets, containing the data involved in the respective stages of the computation. We construct the basic variant of our logic, called PSL, by substituting atoms of Propositional Dynamic Logic (PDL) with queries belonging to a selected query

[1] See http://www.w3.org/2011/prov/wiki/Main_Page

language. The dynamic component, thus inherited from PDL, enables expressing complex provenance patterns, while the embedded queries support access to data artifacts. In the second step, we lift this approach to cater for scenarios in which provenance graphs are themselves described in dedicated provenance ontologies. This way, we obtain the target formalism PSL^M, which, on top of the functionalities offered by PSL, also facilitates the use of a rich metalanguage.

This mechanism is highly independent from the employed representation formalisms, and can be reused in a plug-and-play fashion for a number of combinations of ontology–query languages. Moreover, we demonstrate that PSL^M is computationally well-behaved. By separating the DL-level reasoning tasks from the pure model checking of provenance graphs, we obtain a PTime-completeness result, carried over from the model checking problem in PDL, which remains invariant to the particular choice of the employed ontology/query languages.

Contents. In this work we deliver three main contributions: 1) We introduce PSL^M, a declarative language for expressing complex constraints over data provenance records. 2) By systematically studying the collection of test queries from FPC, mentioned above, we show that PSL^M offers desired modeling capabilities. 3) Finally, we provide a computational analysis of the approach, and report on some satisfying results.

In the remainder of this paper, we first give a short overview of the related work (Section 2) and preliminary notions (Section 3). Next, we incrementally introduce PSL^M (Sections 4, 5) and validate it against the test queries (Section 6). Finally, we study the computational aspects of our framework (Section 7).

2 Related Work

Provenance is nowadays recognized as one of the critical problems to be addressed by the Semantic Web community, attracting increasing interest, e.g. [7]. Existing Semantic Web-based approaches to the problem of verification and querying, such as [8] are persistently technology-driven, and employ combinations of web services, ontologies, triple stores, SPARQL queries, etc. and fail to lay down systematic perspectives on the formal foundations of the problem. Noteworthy exceptions are [9] and [10] which provide, respectively: reproducibility semantics, which are executional in nature, and logic program-based framework for reasoning with a provenance-annotated linked data, where both annotations and data language are specifically restricted. Our paper goes beyond those proposals by providing a cohesive declarative semantic framework based on standard logic and ontology languages, and rich metamodels.

On the formal level, the problem of provenance verification bears a strong resemblance to the traditionally studied verification of transition systems, which in principle encourages the use of similar logic-based techniques [11]. This analogy, however, must be treated with caution. While in usual transition systems states represent complete, propositional abstractions of system's configurations, in the data provenance context states are effectively datasets, reflecting the knowledge of the system in a certain configuration. This creates a need for more expressive verification formalisms, extending the basic program logics, such as PDL [12]. Even Dynamic DLs [13], which are capable of modeling transition systems with states corresponding to DL knowledge bases, are not flexible enough to express rich constraints on the data level. Some other verification formalisms, of a more suitable, data-oriented flavor, have been proposed for verification of data-driven systems [14], knowledge base programs [15], or workflow schemas [16]. However, the central motivation behind their design is to enable representation of all permissable data-altering operations over a fixed data language, with the aim of studying general properties of programs composed of such operations. Consequently, the considered representation languages are strongly restricted in order to ensure decidability of those properties. Such general problems, however, are not of a primary interest in our case, since a provenance record describes by definition a single, completed computation process, which one wants to study ex-post. Hence, rather than abstracting from the richness of a given system and focusing on its possible behaviors, we must enable reasoning machinery which can maximally utilize the available information about the system.

3 Preliminaries

For clarity of exposition, in this paper we consider data represented and managed within the framework of *Description Logics* (DLs), noting that all claims made in this context extend naturally to arbitrary fragments of OWL/RDF(S) languages. In what follows, we briefly recap the preliminaries regarding DLs, and further formalize the notion of data provenance record.

3.1 Description Logics

We use the standard nomenclature and notation for the syntax and semantics of DLs. We refer the reader to [6] for full details. A DL language $\mathcal{L}$ is specified by a vocabulary $\Sigma = (N_C, N_R, N_I)$, where N_C is a set of concept names, N_R a set of role names and N_I a set of individual names, and by a selection of logical operators enabling construction of complex concepts, roles and axioms. Different combinations of operators give rise to DLs of different expressiveness and computational complexity, from the highly expressive $\mathcal{SROIQ}$, underpinning the OWL 2 DL language, to the lightweight $\mathcal{EL}^{++}$ or the DL-*Lite* family, on which tractable OWL profiles are based [17]. For instance, the DL $\mathcal{ALCO}$, a substantial fragment of OWL 2 DL, permits the following constructors:

***Concepts*:**

$$\top \mid \bot \mid A \in N_C \mid \neg C \mid C \sqcap D \mid C \sqcup D \mid \exists r.C \mid \forall r.C \mid \{a\}$$

***Axioms*:**

$$C \sqsubseteq D \mid C(a) \mid r(a, b)$$

where $r \in N_R$, $a, b \in N_I$ and C, D are (possibly complex) concepts.

From the data management perspective, a set of concept inclusions $C \sqsubseteq D$ is typically considered an ontology which provides access to instance data, represented as a set of assertions of the form $A(a), r(a, b)$ [18]. We implicitly abide by this distinction, but for simplicity refer rather to the general notion of a DL knowledge base.

Definition 1 (Knowledge base). *A* knowledge base $\mathcal{K}$ *over a DL language* $\mathcal{L}$ *is a finite set of axioms allowed by the syntax of* $\mathcal{L}$*. The set of all knowledge bases over* $\mathcal{L}$ *is denoted by* $\mathbf{K}(\mathcal{L})$*.*

The semantics of $\mathcal{L}$ is given in terms of the usual model-theoretic interpretations. An interpretation $\mathcal{I}$ is a model of a knowledge base *iff* it satisfies all its axioms. We say that an axiom ϕ is *entailed* by a knowledge base $\mathcal{K}$, denoted as $\mathcal{K} \models \phi$ *iff* ϕ is satisfied in every model $\mathcal{I}$ of $\mathcal{K}$.

Further, we recall the notion of conjunctive queries (CQs) — the most popular class of first-order queries studied in the context of DLs [19]. The problem of answering CQs is known to be decidable for most DLs, and is effectively handled by existing tools, such as DL reasoners (e.g. Pellet) or, as in case of DL-*Lite* family, relational database management systems (e.g. Mastro[2]). Let N_V be a countably infinite set of variables. A *conjunctive query* over a DL language $\mathcal{L}$ with the vocabulary $\Sigma = (N_C, N_R, N_I)$ is a first-order formula $\phi = \exists \vec{y}.q(\vec{x}, \vec{y})$, where $\vec{x}, \vec{y} \in N_V$ are sequences of variables and q is a conjunction of atoms over Σ. The free variables $\vec{x}$ occurring in ϕ are also called the answer variables and denoted as $\mathrm{avar}(\phi)$. For a CQ ϕ, a ϕ-substitution is a mapping $\mu : \mathrm{avar}(\phi) \mapsto N_I$. We write $\mu(\phi)$ to denote the formula resulting from applying μ to ϕ. We call μ a *certain answer* to ϕ w.r.t. a knowledge base $\mathcal{K}$, whenever $\mathcal{K} \models \mu(\phi)$, i.e. whenever $\mu(\phi)$ is satisfied in all models $\mathcal{I}$ of $\mathcal{K}$.

3.2 Data Provenance Records

The definition of a provenance record that we adopt here, and further refine in Section 5, is the simplest abstraction of the proposals currently discussed in the course of a W3C standardization effort. Those proposals, building largely on the specification of the Open Provenance Model [2], consider a provenance record to be a basic graph structure (such as presented in Figure 1) representing the whole documented history of interactions between processes and data artifacts during a certain computation, where data artifacts are in fact datasets (knowledge bases) expressed in DLs. The choice of the OPM foundations for our approach is motivated largely by the fact that OPM is suggested as the intended formalism for representing provenance in the expected W3C recommendation. In principle, however, the level of abstraction which we endorse here goes beyond particular, concrete encodings of provenance information, and builds only on generic provenance notions present also in other formalisms used for recording provenance, such as Proof Markup Language [20,21]. Crucially, our approach generalizes over any (transition) graph-based representation of data provenance.

A *directed graph* is a pair (V, E), where V is a non-empty set of nodes and E is a set of ordered pairs from $V \times V$, called edges. A *bipartite graph* is a graph $(V \cup W, E)$, where $V \cup W$ is a set of nodes and E a set of edges such that

[2] See http://www.dis.uniroma1.it/quonto/

$E \subseteq V \times W \cup W \times V$. An *edge-labeled graph* is a triple (V, E, l), such that (V, E) is a graph and $l : E \mapsto R$ assigns a relation name from a set R to every edge in E. A graph (V, E) is called *acyclic iff* for every node $v \in V$, there exists no sequence $w_1, \ldots, w_n \in V$, such that $(v, w_1), \ldots, (w_{n-1}, w_n), (w_n, v) \in E$.

Definition 2 (Provenance graph). *An* $\mathcal{L}$-provenance graph *is a tuple* $G = (P, D, E, l, k)$*, where* $(P \cup D, E, l)$ *is a bipartite, directed, acyclic, edge-labeled graph, and* k *is a function* $k : D \mapsto \mathbf{K}(\mathcal{L})$*. The nodes in* P *are called* processes *and in* D data artifacts*.*

By convention, we identify process nodes with unique process invocations that occurred during the recorded computation, and data artifact nodes with the corresponding DL knowledge bases $\{k(d) \mid d \in D\}$ that were involved. Note, that we do not presume any specific causal relationships between the represented entities. We are only interested in the formal properties of the graphs.

4 Provenance Specification Logic

Formal verification is the task of checking whether a certain formal structure satisfies the property described by a given formula of a dedicated specification language. The properties of data provenance records which we aim to capture here are essentially complex relationships between the structural patterns occurring in the provenance graphs and the contents of data artifacts. Three typical constraints, representative of most reasoning tasks requested from practical provenance systems [3,4,5], are e.g.:

1. $r(a, b)$ holds in data artifact d_1 and d_1 is reachable via edge *succeeds* from processes p_1 and p_2,
2. a data artifact in which $D(a)$ does not hold is reachable via a finite sequence of two-step edge compositions *wasGeneratedBy*-*used* from a data artifact in which $D(a)$ holds,
3. if $D(a)$ holds in any data artifact related to process p_1 via either $input_1$ or $input_2$, then p_1 must be related via *output* to some data artifact in which $r(a, y)$ holds, for some arbitrary y.

These informally stated properties are clearly satisfied by the respective provenance graphs, illustrated in Figure 2, where nodes p_1, p_2 represent process nodes, and d_1, d_2, d_3 data artifacts, whose contents are listed inside the nodes.

The ability of expressing constraints of this flavor is the key feature of a big family of program verification formalisms based on dynamic logics, in particular the prominent Propositional Dynamic Logic (PDL) [12]. The *provenance specification logic* (PSL), which we introduce below, is a data-oriented extension of PDL. Essentially, we substitute propositional letters of PDL formulas with queries belonging to a certain query language. The dynamic component of PSL enables explicit modeling of requested provenance patterns, while the queries allow for accessing the contents of data artifacts. The choice of an adequate query language is in principle an application-driven decision, depending strongly on

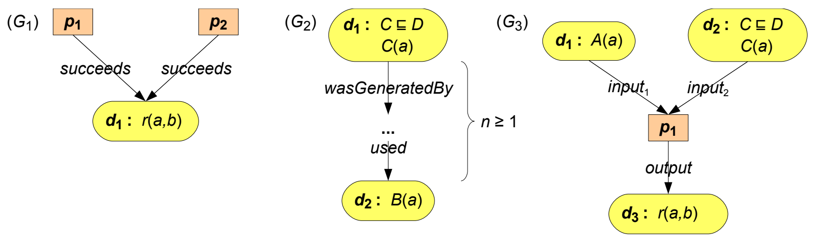


Fig. 2. Sample provenance graphs

the underlying data language. For instance, if data artifacts use RDF(S) representation, a natural candidate is SPARQL [22]. As our focus is on the general DL setup, we consider the class of conjunctive queries, as introduced in Section 3.

Definition 3 (PSL: syntax). *Let $G = (P, D, E, l, k)$ be an $\mathcal{L}$-provenance graph and R the set of relation names used in G. Then the* provenance specification language *over G is the smallest language induced by the following grammar:*

Object queries:

$$\phi \ := \ \textit{CQs over } \mathcal{L}$$

Path expressions:

$$\pi \ := \ r \ \mid \ \pi;\pi \ \mid \ \pi \cup \pi \ \mid \ \pi^- \ \mid \ \pi^* \ \mid \ v? \ \mid \ \alpha?$$

where $r \in R$ and $v \in P \cup D$,

Provenance formulas:

$$\alpha \ := \ \{\phi\} \ \mid \ \top \ \mid \ \langle\pi\rangle\alpha \ \mid \ \alpha \wedge \alpha \ \mid \ \neg\alpha$$

Whenever convenient we use the usual abbreviations $\bot = \neg\top$, $[\pi] = \neg\langle\pi\rangle\neg$, $\alpha \vee \beta = \neg(\neg\alpha \wedge \neg\beta)$ and $\alpha \rightarrow \beta = \neg\alpha \vee \beta$.

Following the CQ notation, by $\mathrm{avar}(\alpha)$ we denote the set of all free (answer) variables occurring in a provenance formula α, i.e. the union of all answer variables from the CQs embedded in α. Note, that different CQs are allowed to share same answer variables. This way one can capture interesting data dependencies between the contents of data artifacts. An α-substitution is a mapping $\mu : \mathrm{avar}(\alpha) \mapsto N_I$ and $\mu(\alpha)$ denotes the result of applying μ to α. We say that $\mu(\alpha)$ is *satisfied* in an $\mathcal{L}$-provenance graph $G = (P, D, E, l, k)$ in a node $v \in P \cup D$ *iff* $G, v \Vdash \mu(\alpha)$, where the *satisfaction relation* $\Vdash$ is defined as follows.

Definition 4 (PSL: semantics). *The* satisfaction relation $\Vdash$ *for PSL formulas is given by a simultaneous induction over the structure of provenance formulas and path expressions,* w.r.t. *a substitution μ. For an $\mathcal{L}$-provenance graph $G = (P, D, E, l, k)$ and every $v, w \in P \cup D$:*

Provenance formulas:

$G, v \Vdash \{\phi\}$ iff *$v \in D$ and $k(v) \models \mu(\phi)$,*
$G, v \Vdash \top$,
$G, v \Vdash \langle\pi\rangle\alpha$ iff *there exists $w \in P \cup D$, s.t. $G \Vdash v \xrightarrow{\pi} w$ and $G, w \Vdash \alpha$,*

$G, v \Vdash \alpha \wedge \beta$ iff $G, v \Vdash \alpha$ *and* $G, v \Vdash \beta$,
$G, v \Vdash \neg\alpha$ iff $G, v \nVdash \alpha$,

Path expressions:

$G \Vdash v \xrightarrow{r} w$ iff $(v, w) \in E$ *and* $l(v, w) = r$,
$G \Vdash v \xrightarrow{\pi;\sigma} w$ iff *there is* $u \in P \cup D$ *s.t.* $G \Vdash v \xrightarrow{\pi} u$ *and* $G \Vdash u \xrightarrow{\sigma} w$,
$G \Vdash v \xrightarrow{\pi\cup\sigma} w$ iff $G \Vdash v \xrightarrow{\pi} w$ *or* $G \Vdash v \xrightarrow{\sigma} w$,
$G \Vdash v \xrightarrow{\pi^-} w$ iff $G \Vdash w \xrightarrow{\pi} v$,
$G \Vdash v \xrightarrow{\pi^*} w$ iff $v(\xrightarrow{\pi})^* w$, *where* $(\xrightarrow{\pi})^*$ *is the transitive reflexive closure of* $\xrightarrow{\pi}$ *on* G,
$G \Vdash v \xrightarrow{v?} v$,
$G \Vdash v \xrightarrow{\alpha?} v$ iff $G, v \Vdash \alpha$.

Observe, that unlike in typical transition systems, only selected nodes in provenance graphs — exactly the data artifacts in D — represent the states over which object queries can be evaluated. Irrespective of this deviation, the model checking problem, underlying formal verification tasks, is defined as usual.

Model Checking 1 (PSL formulas) *Given an $\mathcal{L}$-provenance graph $G = (P, D, E, l, k)$, a node $v \in P \cup D$, a PSL provenance formula α and an α-substitution μ, decide whether $G, v \Vdash \mu(\alpha)$.*

It is easy to check that the following PSL formulas express precisely the properties from the three examples presented in the opening of this section, and are satisfied by the specified graphs, nodes and substitutions (Figure 2):

1. $\alpha := \langle p_1?; \mathit{succeeds}; d_1?\rangle(\{r(x, y)\} \wedge \langle \mathit{succeeds}^-; p_2?\rangle\top)$,
   where $G_1, p_1 \Vdash \mu(\alpha)$ for $\mu = \{x \mapsto a, y \mapsto b\}$,
2. $\alpha := \{D(x)\} \wedge \langle(\mathit{wasGeneratedBy}; \mathit{used})^*\rangle\neg\{D(x)\}$,
   where $G_2, d_1 \Vdash \mu(\alpha)$ for $\mu = \{x \mapsto a\}$,
3. $\alpha := \langle p_1?\rangle(\langle(\mathit{input}_1 \cup \mathit{input}_2)^-\rangle\{D(x)\} \to \langle \mathit{output}\rangle\{\exists y.r(x, y)\})$,
   where $G_3, p_1 \Vdash \mu(\alpha)$ for $\mu = \{x \mapsto a\}$.

For a more practical illustration, we model two use-cases from the eScience domain. The first one illustrates a typical problem of provenance-based validation of an eScience experiment, reported in [4].

Example 1. *A bioinformatician, B, downloads a file containing sequence data from a remote database. B then processes the sequence using an* analysis service*. Later, a reviewer, R, suspects that the sequence may have been a* nucleotide *sequence but processed by a service that can only analyze meaningfully* amino acid *sequences. R determines whether this was the case.*

$$\alpha := \{\exists y.\text{Sequence}(x) \wedge \text{analysis-result}(x, y)\} \to$$
$$[\mathit{output}; \mathit{analysis\text{-}service}?; \mathit{input}](\{\text{Amino-acid}(x)\} \wedge \neg\{\text{Nucleotide}(x)\})$$

Solution: The requested property is satisfied by a graph G if $G, v \Vdash \mu(\alpha)$ for every $v \in P \cup D$ and α-substitution $\mu = \{x \mapsto a\}$, where a is the sequence in question. Naturally, we presume a certain underlying representation model, where e.g. *analysis-service* is the name of the cited service, the result of analysis is given via an axiom of type analysis-result(a, y), etc. For lack of space we do not sketch such models for any of the remaining examples, relying on a proper reconstruction by the reader.

As the second example, we formalize one of the queries from FPC [5].

Example 2. See Q6 in Section 1 (*cf.* Figure 1).

$$\alpha := \{\text{Image}(x)\} \wedge \langle \mathit{wasGeneretedBy}; \mathit{softmean}_{1\ldots n}; \mathit{used}\rangle(\{\exists y.\text{Image}(y)\} \wedge \langle(\mathit{wasGeneratedBy}; \mathit{used})^*; \mathit{wasGeneratedBy}; \mathit{align\text{-}warp}_{1\ldots m}\rangle\top)$$

where $\mathit{softmean}_{1\ldots n} := \mathit{softmean}_1? \cup \ldots \cup \mathit{softmean}_n?$ includes all invocations of *softmean* process in the graph, while $\mathit{align\text{-}warp}_{1\ldots m} := \mathit{align\text{-}warp}_1? \cup \ldots \cup \mathit{align\text{-}warp}_m?$ all invocations of *align warp* with the specified parameter value.

Solution: For every $v \in P \cup D$ and α-substitution μ, if $G, v \Vdash \mu(\alpha)$, then μ is a requested resource.

Observe, that in the latter example not all information requested in the query can be expressed is the PSL formula in a direct, declarative manner. Namely, the selection of *softmean* and *align warp* invocations has to be encoded by an exhaustive enumeration of all the nodes satisfying the specified description. This shortcoming, which affects the high-level modeling capabilities of our formalism, is exactly what motivates the extension introduced in the next section.

5 Provenance Metalanguage

In practice, the relevant provenance information can be much richer than reflected in our abstract notion of provenance graphs. Typically, provenance records account also for the execution context of all processes, including their parametrization, time, responsible actors, etc. [1,2,3]. Moreover, they use complex taxonomies for classifying all these resources. Consequently, the structure of a provenance graph, along the accompanying contextual information, is commonly expressed by means of another DL-based language, used orthogonally to that representing the contents of data artifacts [23,3]. For instance, the provenance graph G implicitly referred to in Example 2, would be likely represented as a knowledge base containing, among others, the axioms listed in Table 1. Formally, we define such a meta-level representation of provenance graphs as follows.

Definition 5 (Metalanguage, metaknowledge base). *Let* $G = (P, D, E, l, k)$ *be an* $\mathcal{L}$*-provenance graph and* R *the set of relation names used in* G*. Let* $\mathcal{L}_G$ *be a DL language with the vocabulary* $\Gamma = (M_C, M_R, M_I)$*, such that* $R \subseteq M_R$ *and* $P \cup D \subseteq M_I$*, and* $\mathcal{K}_G$ *a knowledge base over* $\mathcal{L}_G$*. Then,* $\mathcal{L}_G$ *is called the* metalanguage *and* $\mathfrak{K}_G = (\mathcal{K}_G, k)$ *the* metaknowledge base *over* G iff *the following conditions are satisfied:*

Table 1. A DL knowledge base encoding (part of) a provenance graph

$Artifact \sqsubseteq \neg Process$	$Softmean \sqsubseteq Process$
$Artifact \sqsubseteq \forall wasGeneratedBy.Process$	$Align\text{-}warp \sqsubseteq Process$
$Process \sqsubseteq \forall used.Artifact$	$Align\text{-}warp \sqsubseteq \exists hasArgValue.\mathtt{String}$
$Softmean(softmean_i)$	- for every node $softmean_i$
$Align\text{-}warp(align\text{-}warp_i)$	- for every node $align\text{-}warp_i$
$hasArgValue(align\text{-}warp_i,$ "-m 12"$)$	- for every node $align\text{-}warp_i$ corresponding to an invocation of *align warp* with argument "-m 12"

1. $D = \{v \in M_I \mid \mathcal{K}_G \models Artifact(v)\}$, *for a designated concept* $Artifact \in M_C$,
2. *for every* $r \in R$ *and* $v, w \in M_I$, *it holds that* $(v, w) \in E$ *and* $l(v, w) = r$ iff $\mathcal{K}_G \models r(v, w)$.

Assuming the names in M_I are interpreted uniquely, it is easy to see that the structure of G is isomorphically encoded in the set of role assertions, over role names in R, entailed by $\mathcal{K}_G$. As the positions of data artifacts remain unaltered, one can immediately rephrase the definition of the satisfaction relation $\Vdash$, to show that for any PSL formula α, node $v \in P \cup D$, and an α-substitution μ it is the case that $G, v \Vdash \mu(\alpha)$ *iff* $\mathfrak{K}_G, v \Vdash \mu(\alpha)$. More interestingly, however, we can instead slightly extend the provenance specification language to make a vital use of the newly included meta-level information.

Definition 6 (PSL^M: syntax). *Let* $G = (P, D, E, l, k)$ *be an* $\mathcal{L}$*-provenance graph and* $\mathcal{L}_G$, $\mathfrak{K}_G$ *the metalanguage and the metaknowledge base over* G, *respectively. Then the* provenance specification language (with metalanguage) *over* $\mathfrak{K}_G$ *is the smallest language induced by the grammar of PSL over* G *(Definition 3), modulo the revision of path expressions:*

Path expressions:

$$\pi := r \mid \pi;\pi \mid \pi \cup \pi \mid \pi^- \mid \pi^* \mid v? \mid C? \mid \alpha?$$

where $r \in M_R$, $v \in M_I$ *and* C *is a concept in* $\mathcal{L}_G$,

For a PSL^M provenance formula α, and an α-substitution μ, we say that $\mu(\alpha)$ is *satisfied* a metaknowledge base $\mathfrak{K}_G$ in an instance $v \in M_I$ *iff* $\mathfrak{K}_G, v \Vdash \mu(\alpha)$, where the satisfaction relation $\Vdash$ is defined as follows.

Definition 7 (PSL^M: semantics). *The* satisfaction relation $\Vdash$ *for PSL^M formulas is given by a simultaneous induction over the structure of provenance formulas and path expressions,* w.r.t. *a substitution* μ. *Let* $\mathcal{L}_G$ *be the metalanguage with vocabulary* $\Gamma = (M_C, M_R, M_I)$ *and* $\mathfrak{K}_G = (\mathcal{K}_G, k)$ *the metaknowledge base over an* $\mathcal{L}$*-provenance graph* G. *For all individual names* $v, w \in M_I$:

Provenance formulas:

$\mathfrak{K}_G, v \Vdash \{\phi\}$ iff $\mathcal{K}_G \models Artifact(v)$ *and* $k(v) \models \mu(\phi)$,

$\mathfrak{K}_G, v \Vdash \langle\pi\rangle\alpha$ iff *there exists* $w \in M_I$, *s.t.* $\mathfrak{K}_G \Vdash v \xrightarrow{\pi} w$ *and* $\mathfrak{K}_G, w \Vdash \alpha$,

Path expressions:

$$\mathfrak{K}_G \Vdash v \xrightarrow{r} w \text{ iff } \mathcal{K}_G \models r(v,w),$$
$$\mathfrak{K}_G \Vdash v \xrightarrow{\pi;\sigma} w \text{ iff } \textit{there is } u \in M_I \textit{ s.t. } \mathfrak{K}_G \Vdash v \xrightarrow{\pi} u \textit{ and } \mathfrak{K}_G \Vdash u \xrightarrow{\sigma} w,$$
$$\mathfrak{K}_G \Vdash v \xrightarrow{C?} v \text{ iff } \mathcal{K}_G \models C(v),$$

where the remaining conditions are exactly as in Definition 4 (modulo $G/\mathfrak{K}_G$).

The model checking problem is rephrased accordingly.

Model Checking 2 (PSL^M formulas) *Given a metaknowledge base $\mathfrak{K}_G$ over an $\mathcal{L}$-provenance graph G, an instance $v \in M_I$, a PSL^M provenance formula α, and an α-substitution μ, decide whether $\mathfrak{K}_G, v \Vdash \mu(\alpha)$.*

The usefulness of the presented extension, in particular of the test operator $C?$, which allows for referring to graph nodes generically by their types, inferred from the metaknowledge base, can be observed in the following example.

Example 3. See Q6 in Section 1 and Example 2. We restate the formula α as:

$$\begin{gathered}\alpha := \{\text{Image}(x)\} \wedge \langle \textit{wasGeneretedBy}; \textit{Softmean}?; \textit{used}\rangle \\ (\{\exists y.\text{Image}(y)\} \wedge \langle (\textit{wasGeneratedBy}; \textit{used})^*; \textit{wasGeneratedBy}; \\ (\textit{Align-warp} \sqcap \exists \textit{hasArgValue}.\{\text{“-m 12”}\})?\rangle \top)\end{gathered}$$

where $\mathcal{K}_G$, in the metaknowledge base $\mathfrak{K}_G = (\mathcal{K}_G, k)$, contains (among others) the axioms from Table 1.

Solution: For every $v \in M_I$ and α-substitution μ, if $\mathfrak{K}_G, v \Vdash \mu(\alpha)$, then μ is a requested resource.

Compared to its PSL variant from Example 2, the PSL^M formula used in Example 3 is much more succinct and explicitly represents all requested information. More importantly, thanks to the use of a generic vocabulary for classifying nodes (here: concepts *Softmean*, *Align-warp*$\sqcap\exists$*hasArgValue*.{“-m 12”}), instead of their enumerations, the formula is also more input-independent, in the sense that it can be directly reused to verify/query alternative provenance records obtained from running the same workflows.

6 Evaluation

In order to validate our approach in practical scenarios, we have analyzed the complete list of test queries from The First Provenance Challenge, which to our knowledge constitutes a so far unique ‘golden standard’ for the provenance community. Below we model possible solutions using the logic PSL^M and elaborate on our findings.

Q1. *Find the process that led to Atlas X Graphic / everything that caused Atlas X Graphic to be as it is. This should tell us the new brain images from which the averaged atlas was generated, the warping performed etc.*

$$\alpha_1 := (\top \vee \{\top(x)\}) \wedge \langle (\mathit{used}^- \cup \mathit{wasGeneratedBy}^-)^*; \mathit{Atlas\text{-}X\text{-}Graphic?}\rangle \top$$

Solution: Every $v \in M_I$ and μ such that $\mathfrak{K}, v \Vdash \mu(\alpha_1)$ are requested resources.

Q2. *Find the process that led to Atlas X Graphic, excluding everything prior to the averaging of images with softmean.*

$$\alpha_2 := \alpha_1 \wedge [(\mathit{used}^- \cup \mathit{wasGeneratedBy}^-)^*]\neg\langle \mathit{wasGeneratedBy}^-; \mathit{Softmean?}\rangle \top$$

Solution: Every $v \in M_I$ and μ such that $\mathfrak{K}, v \Vdash \alpha_2$ are requested resources.

Q3. *Find the Stage 3, 4 and 5 details of the process that led to Atlas X Graphic.*

Comment: This is a complex search/verification task, whose reasoning parts can be accomplished by a mix of formulas used in Q1, Q2. Essentially, one must decide what the relevant details are and retrieve them by applying appropriate provenance formulas over the provenance graphs.

Q4. *Find all invocations of procedure align warp using a twelfth order nonlinear 1365 parameter model, i.e. align warp procedure with argument -m 12, that ran on a Monday.*

$$\alpha_4 := \langle (\mathit{Align\text{-}warp} \sqcap \exists \mathit{hasArgValue}.\{\text{“-m 12”}\} \sqcap \exists \mathit{executedOn.Monday})?\rangle \top$$

Solution: Every $v \in M_I$ such that $\mathfrak{K}, v \Vdash \alpha_4$ is a requested resource.

Q5. *Find all Atlas Graphic images outputted from workflows where at least one of the input Anatomy Headers had an entry global maximum=4095. The contents of a header file can be extracted as text using the scanheader AIR utility.*

$$\begin{aligned}\alpha_5 := \langle \mathit{AtlasGraphic?}\rangle(\{\text{Image}(x)\} \wedge \langle (\mathit{wasGeneratedBy}; \mathit{used})^*\rangle \\ \langle \mathit{AnatomyHeader?}\rangle\{\text{hasValue(global-maximum, “4095”)}\})\end{aligned}$$

Solution: For every $v \in M_I$ and μ such that $\mathfrak{K}, v \Vdash \mu(\alpha_5)$, μ is a requested resource.

Q6. Example 3, discussed in the previous section.

Q7. *A user has run the workflow twice, in the second instance replacing each procedures (convert) in the final stage with two procedures: pgmtoppm, then pnmtojpeg. Find the differences between the two workflow runs. The exact level of detail in the difference that is detected by a system is up to each participant.*

Comment: This is a complex search/verification task, whose reasoning parts can be accomplished by posing a number of model checking problems. Essentially, for each relevant provenance formula one must verify it over both graphs and compare the obtained answers.

Q8. *A user has annotated some anatomy images with a key-value pair center=UChicago. Find the outputs of align warp where the inputs are annotated with center=UChicago.*

$$\alpha_8 := \{\top(x)\} \wedge \langle \mathit{wasGeneratedBy}; \mathit{Align\text{-}warp}?; \mathit{used}; \mathit{AnatomyImage}? \rangle \\ \{\exists y.(\mathrm{Image}(y) \wedge \mathrm{center}(y, \mathrm{UChicago}))\}$$

Solution: For every $v \in M_I$ and μ such that $\mathfrak{K}, v \Vdash \mu(\alpha_8)$, μ is a requested resource.

Q9. *A user has annotated some atlas graphics with key-value pair where the key is studyModality. Find all the graphical atlas sets that have metadata annotation studyModality with values speech, visual or audio, and return all other annotations to these files.*

$$\alpha_9 := \langle \mathit{AtlasGraphic}?; \exists \mathit{studyModality}.\{\mathit{speech}\}? \cup \exists \mathit{studyModality}.\{\mathit{visual}\}? \cup \\ \exists \mathit{studyModality}.\{\mathit{radio}\}? \rangle \top$$

Solution: Every $v \in M_I$ such that $\mathfrak{K}, v \Vdash \alpha_9$ is a requested resource. Finding other annotations can be accomplished by posing a number of model checking problems w.r.t. the identified resources.

The above analysis shows that typical reasoning tasks over data provenance records consist of two components: search and logical verification. As far as verification is concerned, the logic PSL^M proves well suited for modeling requested properties and queries. In particular, out of the 9 considered problems, at least 5 — Q1, Q2, Q5, Q6, Q8 — can be solved directly, using a combination of all distinctive features of PSL^M, namely: PDL-like path expressions, embedded CQs and the metalanguage. Queries Q4, Q9 can be answered without the use of embedded CQs. Problems Q3, Q7 and partially Q9 are in fact descriptions of complex search/verification tasks, which can be decomposed into a number of individual verification problems. Those, in turn, can be addressed using PSL^M in the same fashion as in the remaining cases.

7 Reasoning and Complexity

The close relationship of PSL^M to PDL can be conveniently exploited on the computational level. Crucially, PSL^M model checking can be decoupled into two separate problems:

1. construction of a finite-state transition system and a PDL formula (involving polynomially many CQ answering / DL entailment problems),
2. PDL model checking.

This technically unsurprising, but fully intended result has some significant theoretical and practical implications. From the theoretical perspective, it allows for identifying a complexity bound, invariant to the cost of reasoning with the particular DL languages used in the representation. From the practical viewpoint, it opens up a possibility of building simple, yet well-grounded and efficient reasoning architectures based on existing, highly optimized DL reasoners, query engines (e.g. Mastro), and PDL model checkers (e.g. MCPDL[3]).

[3] See `http://www2.tcs.ifi.lmu.de/~axelsson/veri_non_reg/pdlig_mc.html`

In the following, we sketch the reduction procedure. Its full description and the missing proofs are presented in the online technical report [24]. First, recall the notion of a finite-state transition system.

Definition 8 (Transition system). *Let $\mathcal{P} = \{p, q, \ldots\}$ be a set of propositional letters and $\mathcal{A} = \{r, s, \ldots\}$ a set of atomic program names. Then a* finite-state transition system *is a tuple $\mathcal{S} = (W, \{\xrightarrow{r} \mid r \in \mathcal{A}\}, \mathcal{I}\})$, where:*

- *W is a finite, non-empty set of elements called* states*,*
- *$\xrightarrow{r} \subseteq W \times W$ is a* transition relation *corresponding to program r,*
- *$\mathcal{I} : W \mapsto 2^{\mathcal{P}}$ is a propositional* valuation *function.*

Let α be a PSL^M provenance formula, μ an α-substitution, $\mathcal{L}_G$ a metalanguage with vocabulary $\Gamma = (M_C, M_R, M_I)$, and $\mathfrak{K}_G = (\mathcal{K}_G, k)$ a metaknowledge base over an $\mathcal{L}$-provenance graph G. Given this input, we define a finite-state transition system $\mathcal{S}(\mathfrak{K}_G, \mu(\alpha)) = (W, \{\xrightarrow{r} \mid r \in \mathcal{A}\}, \mathcal{I}\})$, essentially, by turning individuals of $\mathcal{K}_G$ into states of the system, structuring the transition relations in the system isomorphically to the corresponding role relationships in $\mathcal{K}_G$, and by encoding all relevant information about the individuals in $\mathcal{K}_G$ in the valuation $\mathcal{I}$ over a designated set of propositional letters corresponding to concepts and CQs. This reduction step involves a polynomial number of decision problems of the form $\mathcal{K}_G \models C(v)$ and $k(v) \models \mu(\phi)$, where $v \in M_I$, C is a concept in $\mathcal{L}_G$ and ϕ is a CQ. Further, we transform the formula $\mu(\alpha)$ by consistently applying substitutions of designated propositions for the corresponding PSL^M subformulas in $\mu(\alpha)$. The resulting expression $\mu(\alpha)^{\mathrm{PDL}}$ is a well-formed PDL formula. Thanks to such "propositionalization" of the input we obtain the following reduction result, where $\mathcal{S}, v \models \varphi$ denotes the model checking problem in PDL, i.e. the problem of deciding whether a PDL formula φ is satisfied in state v of the transition system $\mathcal{S}$.

Theorem 1 (PSL^M vs. PDL). $\mathfrak{K}_G, v \Vdash \mu(\alpha)$ iff $\mathcal{S}(\mathfrak{K}_G, \mu(\alpha)), v \models \mu(\alpha)^{\mathrm{PDL}}$.

It is known that the complexity of model checking in PDL is PTime-complete [12]. Moreover, the size of the transition system $\mathcal{S}(\mathfrak{K}_G, \mu(\alpha))$ and of the formula $\mu(\alpha)^{\mathrm{PDL}}$ is polynomial in $\ell(\mathfrak{K}_G, \alpha, \mu)$, where $\ell(\mathfrak{K}_G, \alpha, \mu)$ is the total size of $\mathfrak{K}_G$, α and μ measured in the number of symbols used. This means, that by disregarding the cost of DL reasoning involved in the construction of $\mathcal{S}(\mathfrak{K}_G, \mu(\alpha))$, we obtain the following time bound.

Theorem 2 (PSL^M model checking: complexity). *Let $\mathfrak{K}_G$ be a metaknowledge base, expressed in $\mathcal{L}_G$, over an $\mathcal{L}$-provenance graph G. Model checking PSL^M formulas over $\mathfrak{K}_G$ is* PTime^{DL}*-complete, where DL is an oracle answering CQs in $\mathcal{L}$ and deciding DL entailment in $\mathcal{L}_G$.*

Finally, we observe that for a given problem $\mathfrak{K}_G, v \Vdash \alpha$ there are at most $2^{\ell(\mathfrak{K}_G, \alpha)}$ different α-substitutions μ, and thus, maximum $2^{\ell(\mathfrak{K}_G, \alpha)}$ different possible pairs $\mathcal{S}(\mathfrak{K}_G, \alpha), \mu(\alpha)^{\mathrm{PDL}}$ to be considered. In practice this number can be dramatically reduced by using smart heuristics to guess only potentially "promising" substitutions. Analogically, the described procedure of constructing the transition systems leaves a considerable space for practical optimizations.

8 Conclusion

In this paper we have introduced the provenance specification logic PSL^M— a dynamic logic-based formalism for verification of data provenance records. The validation, which we have conducted using the test queries of The First Provenance Challenge, shows that a typically requested reasoning task over a data provenance record consist of two components: search and logical verification. As far as the search aspect goes beyond the scope of this work and remains an interesting problem in its own right, requiring smart retrieval and heuristic techniques, we have demonstrated that the logical reasoning part can be successfully captured using the logic and the framework developed here. Moreover, we have shown that the computational cost of performing such tasks is very moderate, and depends mostly on the expressiveness of the languages used for representing the data and the provenance record.

With this contribution, we hope to pave the way towards more systematic and formal studies of the logical problems emerging on the intersection of data-level representations with their meta-level provenance/contextual descriptions, which, in our belief, are of rapidly growing importance in the large-scale, distributed, data- and semantics-rich environments of the Semantic Web and eScience.

Acknowledgments. The authors thank Paul Groth for inspiring discussions on the ideas presented in this paper, and Davide Ceolin for helpful comments.

References

1. Simmhan, Y.L., Plale, B., Gannon, D.: A survey of data provenance in e-science. SIGMOD Rec. 34 (2005)
2. Moreau, L., Clifford, B., Freire, J., Futrelle, J., Gil, Y., Groth, P., Kwasnikowska, N., Miles, S., Missier, P., Myers, J., Plale, B., Simmhan, Y., Stephan, E., den Bussche, J.V.: The open provenance model — core specification (v1.1). Future Generation Computer Systems 27 (2010)
3. Sahoo, S.S., Sheth, A., Henson, C.: Semantic Provenance for eScience: Managing the Deluge of Scientific Data. IEEE Internet Computing 12(4) (2008)
4. Miles, S., Wong, S.C., Fang, W., Groth, P., Zauner, K.P., Moreau, L.: Provenance-based validation of e-science experiments. Web Semantics 5 (2007)
5. Moreau, L., et al.: Special issue: The first provenance challenge. Concurrency and Computation: Practice and Experience 20 (2008)
6. 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)
7. Groth, P., Gil, Y.: Editorial - using provenance in the semantic web. Web Semantics: Science, Services and Agents on the World Wide Web 9(2) (2011)
8. Golbeck, J., Hendler, J.: A semantic web approach to the provenance challenge. Concurrency and Computation: Practice and Experience 20(5) (2008)
9. Moreau, L.: Provenance-based reproducibility in the semantic web. Web Semantics: Science, Services and Agents on the World Wide Web 9(2) (2011)

10. Bonatti, P.A., Hogan, A., Polleres, A., Sauro, L.: Robust and scalable linked data reasoning incorporating provenance and trust annotations. Web Semantics: Science, Services and Agents on the World Wide Web 9(2) (2011)
11. Clarke, E.M., Grumberg, O., Peled, D.A.: Model Checking. The MIT Press (2000)
12. Lange, M.: Model checking propositional dynamic logic with all extras. Journal of Applied Logic 4(1) (2006)
13. Wolter, F., Zakharyaschev, M.: Dynamic description logics. In: Proceedings of AiML 1998 (2000)
14. Vianu, V.: Automatic verification of database-driven systems: a new frontier. In: Proceedings of ICDT 2009 (2009)
15. Calvanese, D., De Giacomo, G., Lenzerini, M., Rosati, R.: Actions and programs over description logic knowledge bases: A functional approach. In: Lakemeyer, G., McIlraith, S.A. (eds.) Knowing, Reasoning, and Acting: Essays in Honour of Hector Levesque. College Publications (2011)
16. Karamanolis, C.T., Giannakopoulou, D., Magee, J., Wheater, S.M.: Model checking of workflow schemas. In: Proceedings of EDOC 2000 (2000)
17. Motik, B.: OWL 2 web ontology language profiles. Technical report, W3C Recommendation (2009), http://www.w3.org/TR/owl2-profiles/
18. Calvanese, D., Giacomo, G.D., Lembo, D., Lenzerini, M., Rosati, R.: Data complexity of query answering in description logics. In: Proceedings of KR 2006 (2006)
19. Glimm, B., Horrocks, I., Lutz, C., Sattler, U.: Conjunctive query answering for the description logic SHIQ. Journal of Artificial Intelligence Research (2007)
20. da Silva, P.P., McGuinness, D.L., Fikes, R.: A proof markup language for semantic web services. Journal of Information Systems - Special Issue: The Semantic Web and Web Services 31(4) (2006)
21. Mcguinness, D.L., Ding, L., Silva, P.P.D., Chang, C.: Pml2: A modular explanation interlingua. In: Proceedings of ExaCt 2007 (2007)
22. Prud'hommeaux, E., Seaborne, A.: SPARQL query language for RDF. Technical report, W3C Recom. (2008), http://www.w3.org/TR/rdf-sparql-query/
23. Belhajjame, K., et al.: The PROV Ontology: Model and formal semantics. Technical report, W3C Draft (2011), http://dvcs.w3.org/hg/prov/raw-file/default/ontology/ProvenanceFormalModel.html
24. Klarman, S., Schlobach, S., Serafini, L.: Formal verification of data provenance records. Technical report (2012), http://klarman.synthasite.com/resources/KlaEtAlISWC12.pdf

Cost Based Query Ordering over OWL Ontologies

Ilianna Kollia[1,2] and Birte Glimm[1]

[1] University of Ulm, Germany
birte.glimm@uni-ulm.de
[2] National Technical University of Athens, Greece
ilianna2@mail.ntua.gr

Abstract. The paper presents an approach for cost-based query planning for SPARQL queries issued over an OWL ontology using the OWL Direct Semantics entailment regime of SPARQL 1.1. The costs are based on information about the instances of classes and properties that are extracted from a model abstraction built by an OWL reasoner. A static and a dynamic algorithm are presented which use these costs to find optimal or near optimal execution orders for the atoms of a query. For the dynamic case, we improve the performance by exploiting an individual clustering approach that allows for computing the cost functions based on one individual sample from a cluster. Our experimental study shows that the static ordering usually outperforms the dynamic one when accurate statistics are available. This changes, however, when the statistics are less accurate, e.g., due to non-deterministic reasoning decisions.

1 Introduction

Query answering—the computation of answers to users' queries w.r.t. ontologies and data—is an important task in the context of the Semantic Web that is provided by many OWL reasoners. Although much effort has been spent on optimizing the 'reasoning' part of query answering, i.e., the extraction of the individuals that are instances of a class or property, less attention has been given to optimizing the actual query answering part when ontologies in expressive languages are used. The SPARQL query language, which was standardized in 2008 by the World Wide Web Consortium (W3C), is widely used for expressing queries in the context of the Semantic Web. We use the OWL Direct Semantics entailment regime of SPARQL 1.1 according to which RDF triples from basic graph patterns are first mapped to extended OWL axioms which can have variables in place of classes, properties and individuals and are then evaluated according to the OWL entailment relation. We focus only on queries with variables in place of individuals since such queries are very common. We call the extended OWL axioms *query atoms* or *atoms*.

In the context of databases or triple stores, cost-based ordering techniques for finding an optimal or near optimal join ordering have been widely applied [12,13]. These techniques involve the maintenance of a set of statistics about relations and indexes, e.g., number of pages in a relation, number of pages in an index, number of distinct values in a column, together with formulas for the estimation of the selectivity of predicates and the estimation of the CPU and I/O costs of query execution that depends amongst others, on the number of pages that have to be read from or written to secondary memory. The formulas for the estimation of selectivities of predicates (result output size of query atoms) estimate the data distributions using histograms, parametric or sampling methods or combinations of them.

In the context of reasoning over ontologies, the formulas should take the cost of executing specific reasoner tasks such as entailment checks or instance retrievals into account. The precise estimation of this cost before query evaluation is difficult as this cost takes values from a wide range. For example, the description logic $\mathcal{SROIQ}$, which underpins the OWL 2 DL standard, has a worst case complexity of 2-NExpTime [6] and typical implementations are not worst case optimal. The hypertableau algorithm that we use has a worst-case complexity of 3-NExpTime in the size of the ontology [9,6].[1]

In this paper we address the optimization task of ordering the query atoms. The optimization goal is to find the execution plan (an order for the query atoms) which leads to the most efficient execution of the query by minimizing the number of needed reasoning tasks and the size of intermediate results. The execution plan which satisfies the above property is determined by means of a cost function that assigns costs to atoms within an execution plan. This cost function is based on heuristics and summaries for statistics about the data, which are extracted from an OWL reasoner model. We explore static and dynamic algorithms together with cluster based sampling techniques that greedily explore the execution plan search space to determine an optimal or near optimal execution plan. Static ordering refers to the finding of a join order before query evaluation starts whereas dynamic ordering determines the ordering of query atoms during query evaluation, taking advantage of already computed query atom results.

2 Preliminaries

In this section we briefly present an overview of the model building tableau and hypertableau calculi and give a brief introduction to SPARQL queries. For brevity, we use the description logic (DL) [1] syntax for examples.

Checking whether an individual s_0 (pair of individuals $\langle s_0, s_1\rangle$) is an instance of a class C (property R) w.r.t. an ontology $\mathcal{O}$ is equivalent to checking whether the class assertion $\neg C(c_0)$ (the negation of the class assertion $C(s_0)$) or the property assertion $(\forall R.\neg\{s_1\})(s_0)$ (s_0 is only R-related to individuals that are not s_1) is inconsistent w.r.t. $\mathcal{O}$. To check this, most OWL reasoners use a model construction calculus such as tableau or hypertableau. In the remainder, we focus on the hypertableau calculus [9], but a tableau calculus could equally be used and we state how our results can be transferred to tableau calculi. The hypertableau calculus starts from an initial set of assertions and, by applying derivation rules, it tries to construct (an abstraction of) a model of $\mathcal{O}$. Derivation rules usually add new class and property assertion axioms, they may introduce new individuals, they can be nondeterministic, leading to the need to choose between several alternative assertions to add or they can lead to a clash when a contradiction is detected. To show that an ontology $\mathcal{O}$ is (in)consistent, the hypertableau calculus constructs a *derivation*, i.e., a sequence of sets of assertions $\mathcal{A}_0, \ldots, \mathcal{A}_n$, such that $\mathcal{A}_0$ contains all assertions in $\mathcal{O}$, $\mathcal{A}_{i+1}$ is the result of applying a derivation rule to $\mathcal{A}_i$ and $\mathcal{A}_n$ is the final

[1] The 2-NExpTime result for $\mathcal{SHOIQ}$+ increases to 3-NExpTime when adding role chains [6].

set of assertions where no more rules are applicable. If a derivation exists such that A_n does not contain a clash, then $\mathcal{O}$ is consistent and A_n is called a *pre-model* of $\mathcal{O}$. Otherwise $\mathcal{O}$ is inconsistent. Each assertion in a set of assertions A_i is derived either deterministically or nondeterministically. An assertion is *derived deterministically* if it is derived by the application of a deterministic derivation rule from assertions that were all derived deterministically. Any other derived assertion is *derived nondeterministically*. It is easy to know whether an assertion was derived deterministically or not because of the dependency directed backtracking that most (hyper)tableau reasoners employ. In the pre-model, each individual s_0 is assigned a label $\mathcal{L}(s_0)$ representing the classes it is (non)deterministically an instance of and each pair of individuals $\langle s_0, s_1\rangle$ is assigned a label $\mathcal{L}(\langle s_0, s_1\rangle)$ representing the properties through which individual s_0 is (non)deterministically related to individual s_1.

The WHERE clause of a SPARQL query consists of graph patterns. Basic graph patterns (BGPs) can be composed to more complex patterns using operators such as UNION and OPTIONAL for alternative and optional selection criteria. The evaluation of (complex) graph patterns is done by evaluating each BGP separately and combining the results of the evaluation. We only consider the evaluation of BGPs since this is the only thing that is specific to a SPARQL entailment regime. We further focus on conjunctive instance queries, i.e., BGPs that retrieve tuples of individuals, which are instances of the queried classes and properties. Such BGPs are first mapped to OWL class and (object) property assertions that allow for variables in place of individuals [7]. For brevity, we directly write mapped BGPs in DL syntax extended to allow for variables. We use the term *query* in the remainder for such BGPs. W.l.o.g., we assume that queries are connected [2].

Definition 1. *Let $\mathcal{O}$ be an ontology with signature $S_\mathcal{O} = (C_\mathcal{O}, R_\mathcal{O}, I_\mathcal{O})$, i.e., $S_\mathcal{O}$ consists of all class, property and individual names occurring in $\mathcal{O}$. We assume that $S_\mathcal{O}$ also contains OWL's special classes and properties such as* owl:Thing. *Let V be a countably infinite set of variables disjoint from $C_\mathcal{O}$, $R_\mathcal{O}$ and $I_\mathcal{O}$. A* term *t is an element from $V \cup I_\mathcal{O}$. Let $C \in C_\mathcal{O}$ be a class, $R \in R_\mathcal{O}$ a property, and $t, t' \in I_\mathcal{O} \cup V$ terms. A* class atom *is an expression $C(t)$ and a* property atom *is an expression $R(t, t')$. A* query *q is a non-empty set of atoms. We use $Vars(q)$ to denote the set of variables and $Inds(q)$ for the set of individuals occurring in q and set $Terms(q) = Vars(q) \cup Inds(q)$. We use $|q|$ to denote the number of atoms in q.*

Let $q = \{at_1, \ldots, at_n\}$ be a query. A mapping *μ* for *q* over *$\mathcal{O}$ is a total function $\mu\colon Terms(q) \to I_\mathcal{O}$ such that $\mu(a) = a$ for each $a \in Inds(q)$. The set Γ_q of all possible mappings for q is defined as $\Gamma_q := \{\mu \mid \mu$ is a mapping for $q\}$. A* solution mapping *μ* for *q* over *$\mathcal{O}$ is a mapping such that $\mathcal{O} \models C(\mu(t))$ for each class atom $C(t) \in q$ and $\mathcal{O} \models R(\mu(t), \mu(t'))$ for each property atom $R(t, t') \in q$.*

3 Extracting Individual Information from Reasoner Models

The first step in the ordering of query atoms is the extraction of statistics, exploiting information generated by reasoners. We use the labels of an initial pre-model to provide us with information about the classes the individuals belong to or the properties in

Algorithm 1. initializeKnownAndPossibleClassInstances

```
Require: a consistent $\mathcal{SROIQ}$ ontology $\mathcal{O}$ to be queried
Ensure: sets $K[C]$ ($P[C]$) of known (possible) instances for each class C of $\mathcal{O}$ are computed
 1: $A_n := buildModelFor(\mathcal{O})$
 2: for all $ind \in \mathsf{I}_{\mathcal{O}}$ do
 3:     for all $C \in \mathcal{L}_{A_n}(ind)$ do
 4:         if $C$ was derived deterministically then
 5:             $K[C] := K[C] \cup \{ind\}$
 6:         else
 7:             $P[C] := P[C] \cup \{ind\}$
 8:         end if
 9:     end for
10: end for
```

which they participate. We exploit this information similarly as was suggested for determining known (non-)subsumers for classes during classification [3]. In the hypertableau calculus, the following two properties hold for each ontology $\mathcal{O}$ and each constructed pre-model A_n for $\mathcal{O}$:

(P1) for each class name C (property R), each individual s_0 (pair of individuals $\langle s_1, s_2\rangle$) in A_n, if $C \in \mathcal{L}_{A_n}(s_0)$ ($R \in \mathcal{L}_{A_n}(\langle s_1, s_2\rangle)$) and the assertion $C(s_0)$ ($R(s_1, s_2)$) was derived deterministically, then it holds $\mathcal{O} \models C(s_0)$ ($\mathcal{O} \models R(s_1, s_2)$).

(P2) for an arbitrary individual s_0 in A_n (pair of individuals $\langle s_1, s_2\rangle$ in A_n) and an arbitrary class name C (simple property R), if $C \notin \mathcal{L}_{A_n}(s_0)$ ($R \notin \mathcal{L}_{A_n}(\langle s_1, s_2\rangle)$), then $\mathcal{O} \not\models C(s_0)$ ($\mathcal{O} \not\models R(s_1, s_2)$).

The term simple property refers to a property that is neither transitive nor has a transitive subproperty.

We use these properties to extract information from the pre-model of a satisfiable ontology $\mathcal{O}$ as outlined in Algorithm 1. In our implementation we use a more complicated procedure to only store the direct types of each individual. The information we extract involves the maintenance of the sets of known and possible instances for all classes of $\mathcal{O}$. The *known instances* of a class C ($K[C]$) are the individuals that can be safely considered instances of the class according to the pre-model, i.e., for each individual $i \in K[C]$, $C(i)$ was derived deterministically in the pre-model. Similarly, the *possible instances* of a class C ($P[C]$) are the individuals i such that $C(i)$ was derived nondeterministically. These possible instances require costly consistency checks in order to decide whether they are real instances of the class.

For simple properties, a procedure to find the known and possible instances of a property or, given an individual, the known and possible property successors or predecessors can be defined similarly. For non-simple properties, $\mathcal{O}$ is expanded with additional axioms that capture the semantics of the transitive relations before the *buildModelFor* procedure is applied since (hyper)tableau reasoners typically do not deal with transitivity directly [9]. In particular, for each individual i and non-simple property R, new classes C_i and C_i^R are introduced and the axioms $C_i(i)$ and $C_i \sqsubseteq \forall R.C_i^R$ are added to $\mathcal{O}$. The consequent application of the transitivity encoding [9] produces axioms that propagate C_i^R to each individual s that is reachable from i via an R-chain. The known and possible R-successors for i can then be determined from the C_i^R instances.

The technique presented in this paper can be used with any (hyper)tableau calculus for which properties (P1) and (P2) hold. All (hyper)tableau calculi used in practice that we are aware of satisfy property (P1). Pre-models produced by tableau algorithms as presented in the literature also satisfy property (P2); however, commonly used optimizations, such as lazy unfolding, can compromise property (P2), which we illustrate with the following example. Let us assume we have an ontology $\mathcal{O}$ containing the axioms $A \sqsubseteq \exists R.(C \sqcap D)$, $B \equiv \exists R.C$ and $A(a)$. It is obvious that in this ontology A is a subclass of B (hence $\mathcal{O} \models B(a)$) since every individual that is R-related to an individual that is an instance of the intersection of C and D is also R-related to an individual that is an instance of the class C. However, even though the assertion $A(a)$ occurs in the ABox, the assertion $B(a)$ is not added in the pre-model when we use lazy unfolding. With lazy unfolding, instead of treating $B \equiv \exists R.C$ as two disjunctions $\neg B \sqcup \exists R.C$ and $B \sqcup \forall R.(\neg C)$ as is typically done for complex class inclusion axioms, B is only lazily unfolded into its definition $\exists R.C$ once B occurs in the label of an individual. Thus, although $(\exists R.(C \sqcap D))(a)$ would be derived, this does not lead to the addition of $B(a)$.

Nevertheless, most (if not all) implemented calculi produce pre-models that satisfy at least the following weaker property:

(P3) for an arbitrary individual s_0 in A_n and an arbitrary class C where C is primitive in $\mathcal{O}$,[2] if $C \notin \mathcal{L}_{A_n}(s_0)$, then $\mathcal{O} \not\models C(s_0)$.

Hence, properties (P2) and (P3) can be used to extract (non-)instance information from pre-models. For tableau calculi that only satisfy (P3), Algorithm 1 can be modified accordingly. In particular, for each non-primitive class C in $\mathcal{O}$ we need to add to P[C] the individuals in $\mathcal{O}$ that do not include the class C in their label.

The proposed technique for determining known and possible instances of classes and properties can be used in the same way with both tableau and hypertableau reasoners. Since tableau algorithms often introduce more nondeterminism than hypertableau, one might, however, find less deterministic derivations, which results in less accurate statistics.

3.1 Individual Clustering

In this section, we describe the procedure for creating clusters of individuals within an ontology $\mathcal{O}$ using the constructed pre-model A_n of $\mathcal{O}$. Two types of clusters are created: class clusters and property clusters. Class clusters contain individuals having the same classes in their label and property clusters contain individuals with the same class and property labels. Property clusters are divided into three categories, those that are based on the first individual of property instances, those based on the second individual and those based on both individuals. First we define, for an ontology $\mathcal{O}$ with pre-model A_n, the relations P_1 and P_2 that map an individual a from $\mathcal{O}$ to the properties for which a has at least one successor or predecessor, respectively:

[2] A class C is considered primitive in $\mathcal{O}$ if $\mathcal{O}$ is unfoldable [14] and it contains no axiom of the form $C \equiv E$.

$$P_1(a) = \{R \mid \exists b \in I_O \text{ such that } R \in \mathcal{L}_{A_n}(\langle a, b\rangle)\}$$
$$P_2(a) = \{R \mid \exists b \in I_O \text{ such that } R \in \mathcal{L}_{A_n}(\langle b, a\rangle)\}$$

Based on these relations, we partition I_O into class clusters $CC = \{C^1, \dots, C^n\}$, property successor clusters $PC_1 = \{C_1^1, \dots, C_1^n\}$, property predecessor clusters $PC_2 = \{C_2^1, \dots, C_2^n\}$ and $I_O \times I_O$ into property clusters $PC_{12} = \{C_{12}^1, \dots, C_{12}^n\}$ such that the clusters satisfy:

$$\forall C \in CC.(\forall a_1, a_2 \in C.(\mathcal{L}_{A_n}(a_1) = \mathcal{L}_{A_n}(a_2)))$$
$$\forall C \in PC_1.(\forall a_1, a_2 \in C.(\mathcal{L}_{A_n}(a_1) = \mathcal{L}_{A_n}(a_2) \text{ and } P_1(a_1) = P_1(a_2)))$$
$$\forall C \in PC_2.(\forall a_1, a_2 \in C.(\mathcal{L}_{A_n}(a_1) = \mathcal{L}_{A_n}(a_2) \text{ and } P_2(a_1) = P_2(a_2)))$$
$$\forall C \in PC_{12}.(\forall \langle a_1, a_2\rangle, \langle a_3, a_4\rangle \in C.(\mathcal{L}_{A_n}(a_1) = \mathcal{L}_{A_n}(a_3), \mathcal{L}_{A_n}(a_2) = \mathcal{L}_{A_n}(a_4) \text{ and } \mathcal{L}_{A_n}(\langle a_1, a_2\rangle) = \mathcal{L}_{A_n}(\langle a_3, a_4\rangle)))$$

4 Query Answering and Query Atom Ordering

In this section, we describe two different algorithms (a static and a dynamic one) for ordering the atoms of a query based on some costs and then we deal with the formulation of these costs. We first introduce the abstract graph representation of a query q by means of a labeled graph G_q on which we define the computed statistical costs.

Definition 2. *A* query join graph G_q *for a query q is a tuple (V, E, E_L), where*

- *$V = q$ is a set of vertices (one for each query atom);*
- *$E \subseteq V \times V$ is a set of edges such that $\langle at_1, at_2\rangle \in E$ iff $Vars(at_1) \cap Vars(at_2) \neq \emptyset$ and $at_1 \neq at_2$;*
- *E_L is a function that assigns a set of variables to each $\langle at_1, at_2\rangle \in E$ such that $E_L(at_1, at_2) = Vars(at_1) \cap Vars(at_2)$.*

In the remainder, we use a, b for individual names, x, y for variables, q for a query $\{at_1, \dots, at_n\}$ with query join graph G_q, and Ω_q for the solution mappings of q.

Our goal is to find a query execution plan, which determines the evaluation order for atoms in q. Since the number of possible execution plans is of order $|q|!$, the ordering task quickly becomes impractical. In the following, we focus on greedy algorithms for determining an execution order, which prune the search space considerably. Roughly speaking, we proceed as follows: We define a cost function, which consists of two components (i) an estimate for the reasoning costs and (ii) an estimate for the intermediate result size. Both components are combined to induce an order among query atoms. In this paper, we simply build the sum of the two cost components, but different combinations such as a weighted sum of the two values could also be used. For the query plan construction we distinguish *static* from *dynamic planning*. For the former, we start constructing the plan by adding a minimal atom according to the order. Variables from this atom are then considered bound, which changes the cost function and might induce a different order among the remaining query atoms. Considering the updated order, we again select the minimal query atom that is not yet in the plan and update the costs. This process continues until the plan contains all atoms. Once a complete plan has been determined the atoms are evaluated. The dynamic case differs in that after selecting an atom for the plan, we immediately determine the solutions for the chosen atom, which are then used to update the cost function. While this yields accurate cost estimates, it can be very costly when all solutions are considered for updating the cost function. Sampling techniques can be used to only test a subset of the solutions, but we show in Section 5 that random sampling, i.e., randomly choosing a percentage of the individuals from the so far computed solutions, is not adequate. For this reason, we propose an alternative sampling approach that is based on the use of the previously described individual clusters. We now make the process of query plan construction more precise, but we leave the exact details of defining the cost function and the ordering it induces to later.

Definition 3. *A* static (dynamic) cost function w.r.t. *q is a function* $s\colon q \times 2^{Vars(q)} \to \mathbb{R} \times \mathbb{R}$ *(*$d\colon q \times 2^{\Gamma_q} \to \mathbb{R} \times \mathbb{R}$*). The two costs are combined to yield a* static ordering $\preceq_s$ *(*dynamic ordering $\preceq_d$*), which is a total order over the atoms of* q*.*

An execution plan *for q is a duplicate-free sequence of query atoms from q. The* initial execution plan *is the empty sequence and a* complete execution plan *is a sequence containing all atoms of q. For* $P_i = (at^1, \ldots, at^i)$ *with* $i < n$ *an execution plan for q with query join graph* $G_q = (V, E, E_L)$*, we define the* potential next atoms q_i for P_i w.r.t. G_q *as* $q_i = q$ *for* P_i *the initial execution plan and* $q_i = \{at \mid \langle at', at\rangle \in E, at' \in \{at^1, \ldots, at^i\}, at \in q \setminus \{at^1, \ldots, at^i\}\}$ *otherwise. The static (dynamic) ordering induces an execution plan* $P_{i+1} = (at^1, \ldots, at^i, at^{i+1})$ *with* $at^{i+1} \in q_i$ *and* $at^{i+1} \preceq_s at$ *(*$at^{i+1} \preceq_d at$*) for each* $at \in q_i$ *such that* $at \neq at^{i+1}$*.*

For $i > 0$, the set of potential next atoms only contains atoms that are connected to an atom that is already in the plan since unconnected atoms will cause an unnecessary blowup of the number of intermediate results. Let $P_i = (at_1, \ldots, at_i)$ with $i \leq n$ be an execution plan for q. The procedure to find the solution mappings Ω_i for P_i is recursively defined as follows: Initially, our solution set contains only the identity mapping $\Omega_0 = \{\mu_0\}$, which does not map any variable to any value. Assuming that we have evaluated the sequence $P_{i-1} = (at_1, \ldots, at_{i-1})$ and we have found the set of solution mappings Ω_{i-1}, in order to find the solution mappings Ω_i of P_i, we use the instance retrieval tasks of reasoners to extend the mappings in Ω_{i-1} to cover the new variables of at_i or the entailment check service of reasoners if at_i does not contain new variables. A detailed description of the method for evaluating a query atom together with optimizations can be found in our previous work [7].

We now define the cost functions s and d more precisely, which estimate the cost of the required reasoner operations (first component) and the estimated result output size (second component) of evaluating a query atom. The intuition behind the estimated value of the reasoner operation costs is that the evaluation of possible instances is much more costly than the evaluation of known instances since possible instances require expensive consistency checks whereas known instances require cheap cache lookups. The estimated result size takes into account the number of known and possible instances and the probability that possible instances are actual instances. Apart from the relations $K[C]$ and $P[C]$ ($K[R]$ and $P[R]$) for the known and possible instances of a class C (property R) from Section 3, we use the following auxiliary relations:

Definition 4. *Let R be a property and a an individual. We define* $\mathsf{sucK}[R]$ *and* $\mathsf{preK}[R]$ *as the set of individuals with known R-successors and R-predecessors, respectively:*

$$\mathsf{sucK}[R] := \{i \mid \exists j.\langle i, j\rangle \in K[R]\} \qquad \textit{and} \qquad \mathsf{preK}[R] := \{i \mid \exists j.\langle j, i\rangle \in K[R]\}.$$

Similarly, we define $\mathsf{sucK}[R, a]$ *and* $\mathsf{preK}[R, a]$ *as the known R-successors of a and the known R-predecessors of a, respectively:*

$$\mathsf{sucK}[R, a] := \{i \mid \langle a, i\rangle \in K[R]\} \qquad \textit{and} \qquad \mathsf{preK}[R, a] := \{i \mid \langle i, a\rangle \in K[R]\}.$$

We analogously define the functions $\mathsf{sucP}[R]$*,* $\mathsf{preP}[R]$*,* $\mathsf{sucP}[R, a]$*, and* $\mathsf{preP}[R, a]$ *by replacing $P[C]$ and $P[R]$ with $K[C]$ and $K[R]$, respectively. We write C_L to denote the cost of a cache lookup in the internal structures of the reasoner, C_E for the cost of an entailment check, and P_{IS} for the possible instance success, i.e, the estimated percentage of possible instances that are actual instances.*

The costs C_L and C_E are determined by recording the average time of previously performed lookups and entailment checks for the queried ontology, e.g., during the initial consistency check, classification, or for previous queries. In the case of C_E, we multiply this number with the depth of the class (property) hierarchy since we only store the direct types of each individual (properties in which each individual participates) and, in order to find the instances of a class (property), we may need to check all its subclasses (subproperties) that contain possible instances. The time needed for an entailment check can change considerably between ontologies and even within an ontology (depending on the involved classes, properties, and individuals). Thus, the use of a single constant for the entailment cost is not very accurate, however, the definition of different entailment costs before executing the reasoning task is very difficult.

The possible instance success, P_{IS}, was determined by testing several ontologies and checking how many of the initial possible instances were real ones, which was around 50% in nearly all ontologies.

4.1 The Static and Dynamic Cost Functions

The static cost function s takes two components as input: a query atom and a set containing the variables of the query atom that are considered bound. The function returns a pair of real numbers for the reasoning cost and result size for the query atom.

Initially, all variables are unbound and we use the number of known and possible instances or successors/predecessors to estimate the number of required lookups and consistency checks for evaluating the query atom and for the resulting number of mappings. For an input of the form $\langle C(x), \emptyset\rangle$ or $\langle R(x, y), \emptyset\rangle$ the resulting pair of real numbers for the computational cost and the estimated result size is computed as

$$\langle |K[at]| \cdot C_L + |P[at]| \cdot C_E, |K[at]| + P_{IS} \cdot |P[at]|\rangle,$$

where at denotes the predicate of the query atom (C or R). If the query atom is a property atom with a constant in the first place, i.e., the input to the cost function is of the form $\langle R(a, x), \emptyset\rangle$, we use the relations for known and possible successors to estimate the computational cost and result size:

$$\langle|\mathsf{sucK}[R, a]| \cdot C_L + |\mathsf{sucP}[R, a]| \cdot C_E, |\mathsf{sucK}[R, a]| + P_{IS} \cdot |\mathsf{sucP}[R, a]|\rangle.$$

Analogously, we use preK and preP instead of sucK and sucP for an input of the form $\langle R(x, a), \emptyset\rangle$. Finally, if the atom contains only constants, i.e., the input to the cost function is of the form $\langle C(a), \emptyset\rangle$, $\langle R(a, b), \emptyset\rangle$, the function returns $\langle C_L, 1\rangle$ if the individual is a known instance of the class or property, $\langle C_E, P_{IS}\rangle$ if the individual is a possible instance and $\langle C_L, 0\rangle$ otherwise, i.e., if the individual is a known non-instance.

After determining the cost of an initial query atom, at least one variable of a consequently considered atom is bound, since during the query plan construction we move over atoms sharing a common variable and we assume that the query is connected. We now define the cost functions for atoms with at least one variable bound. We make the assumption that atoms with unbound variables are more costly to evaluate than atoms with all their variables bound. For a query atom $R(x, y)$ with only x bound, i.e., function inputs of the form $\langle R(x, y), \{x\}\rangle$, we use the average number of known and possible successors of the property to estimate the computational cost and result size:

$$\left\langle \frac{|K[R]|}{|\mathsf{sucK}[R]|} \cdot C_L + \frac{|P[R]|}{|\mathsf{sucP}[R]|} \cdot C_E, \frac{|K[R]|}{|\mathsf{sucK}[R]|} + \frac{|P[R]|}{|\mathsf{sucP}[R]|} \cdot P_{IS} \right\rangle$$

In case only y in $R(x, y)$ is bound, we use the predecessor functions preK and preP instead of sucK and sucP. Note that we now work with an estimated average number of successors (predecessors) for *one individual*.

For the remaining cases (atoms with all their variables bound), we use formulas that are comparable to the ones above for an initial plan, but normalized to estimate the values for one individual. The normalization is important for achieving compatibility with the formulas described above for inputs of the form $\langle R(x, y), \{x\}\rangle$ and $\langle R(x, y), \{y\}\rangle$. For an input query atom of the form $C(x)$ with x a bound variable we use

$$\left\langle \frac{|K[C]|}{|I_O|} \cdot C_L + \frac{|P[C]|}{|I_O|} \cdot C_E, \frac{|K[C]| + P_{IS} \cdot |P[C]|}{|I_O|} \right\rangle$$

Such a simple normalization is not always accurate, but leads to good results in most cases as we show in Section 5. Similarly, we normalize the formulae for property atoms of the form $R(x, y)$ such that $\{x, y\}$ is the set of bound variables of the atom. The two cost components for these atoms are computed as

$$\left\langle \frac{|K[R]|}{|I_O|} \cdot C_L + \frac{|P[R]|}{|I_O|} \cdot C_E, \frac{|K[R]| + P_{IS} \cdot |P[R]|}{|I_O| \cdot |I_O|} \right\rangle$$

For property atoms with a constant and a bound variable, i.e., atoms of the form $R(a, x)$ ($R(x, a)$) with x a bound variable, we use $\mathsf{sucK}[R, a]$ and $\mathsf{sucP}[R, a]$ ($\mathsf{preK}[R, a]$ and $\mathsf{preP}[R, a]$) instead of $K[R]$ and $P[R]$ in the above formulae.

The dynamic cost function d is based on the static function s, but only uses the first equations, where the atom contains only unbound variables or constants. The function takes a pair $\langle at, \Omega\rangle$ as input, where at is a query atom and Ω is the set of solution

Table 1. Query Ordering Example

	Atom Sequences	Known Instances	Possible Instances	Real from Possible Instances
1	C(x)	200	350	200
2	R(x,y)	200	200	50
3	D(y)	700	600	400
4	R(x,y), C(x)	100	150	100
5	R(x,y), D(y)	50	50	40
6	R(x,y), D(y), C(x)	45	35	25
7	R(x,y), C(x), D(y)	45	40	25

mappings for the atoms that have already been evaluated, and returns a pair of real numbers using matrix addition as follows:

$$d(at, \Omega) = \sum_{\mu \in \Omega} s(\mu(at), \emptyset)$$

When sampling techniques are used, we compute the costs for each of the potential next atoms for an execution plan by only considering one individual of each relevant cluster. Which cluster is relevant depends on the query atom for which we compute the cost function and the previously computed bindings. For instance, if we compute the cost of a property atom $R(x, y)$ and we already determined bindings for x, we use the property successor cluster PC_1. Among the x bindings, we then just check the cost for one binding per cluster and assign the same cost to all other x bindings of the same cluster.

A motivating example showing the difference between static and dynamic ordering and justifying why dynamic ordering can be beneficial in our setting is shown below. Let us assume that a query q consists of the three query atoms: $C(x)$, $R(x, y)$, $D(y)$. Table 1 gives information about the known and possible instances of these atoms within a sequence. In particular, the first column enumerates possible execution sequences $S_i = (at_1, \ldots, at_i)$ for the atoms of q. Column 2 (3) gives the number of mappings to known (possible) instances of at_i (i.e., the number of known (possible) instances of at_i) that satisfy at the same time the atoms $(at_1, \ldots, at_{i-1})$. Column 4 gives the number of possible instances of at_i from Column 3 that are real instances (that belong to Ω_i). Let us assume that we have 10,000 individuals in our ontology $\mathcal{O}$. We will now explain via the example what the formulas described above are doing. We assume that $C_L \leq C_E$ which is always the case since a cache lookup is less expensive than a consistency check. In both techniques (static and dynamic) the atom $R(x, y)$ will be chosen first since it has the least number of possible instances (200) while it has the same (or smaller) number of known instances (200) as the other atoms:

$$\begin{aligned} s(R(x, y), \emptyset) &= d(R(x, y), \{\mu_0\}) = \langle 200 \cdot C_L + 200 \cdot C_E, 200 + P_{IS} \cdot 200 \rangle, \\ s(C(x), \emptyset) &= d(C(x), \{\mu_0\}) = \langle 200 \cdot C_L + 350 \cdot C_E, 200 + P_{IS} \cdot 350 \rangle, \\ s(D(y), \emptyset) &= d(D(y), \{\mu_0\}) = \langle 700 \cdot C_L + 600 \cdot C_E, 700 + P_{IS} \cdot 600 \rangle. \end{aligned}$$

In the case of static ordering, the atom $C(x)$ is chosen after $R(x, y)$ since C has less possible (and known) instances than D (350 versus 600):

$$s(C(x), \{x\}) = \left\langle \frac{200}{10,000} \cdot C_L + \frac{350}{10,000} \cdot C_E, \frac{200 + 350 \cdot P_{IS}}{10,000} \right\rangle,$$
$$s(D(y), \{y\}) = \left\langle \frac{700}{10,000} \cdot C_L + \frac{600}{10,000} \cdot C_E, \frac{700 + 600 \cdot P_{IS}}{10,000} \right\rangle.$$

Hence, the order of evaluation in this case will be $P = (R(x, y), C(x), D(y))$ leading to 200 (row 2) + 150 (row 4) + 40 (row 7) entailment checks. In the dynamic case, after the evaluation of $R(x, y)$, which gives a set of solutions Ω_1, the atom $D(y)$ has fewer known and possible instances (50 known and 50 possible) than the atom $C(x)$ (100 known and 150 possible) and, hence, a lower cost:

$$d(D(y), \Omega_1) = \langle 50 \cdot C_L + 150 \cdot C_L + 50 \cdot C_E, 50 + 0 + 50 \cdot P_{IS} \rangle,$$
$$d(C(x), \Omega_1) = \langle 100 \cdot C_L + 0 \cdot C_L + 150 \cdot C_E, 100 + 0 + 150 \cdot P_{IS} \rangle.$$

Note that applying a solution $\mu \in \Omega_1$ to $D(y)$ ($C(x)$) results in a query atom with a constant in place of y (x). For $D(y)$, it is the case that out of the 250 R-instances, 200 can be handled with a look-up (50 turn out to be known instances and 150 turn out not to be instances of D), while 50 require an entailment check. Similarly, when considering $C(x)$, we need 100 lookups and 150 entailment checks. Note that we assume the worst case in this example, i.e., that all values that x and y take are different. Therefore, the atom $D(y)$ will be chosen next leading to the execution of the query atoms in the order $P = (R(x, y), D(y), C(x))$ and the execution of 200 (row 2) + 50 (row 5) + 35 (row 6) entailment checks.

5 Evaluation

We tested our ordering techniques with the Lehigh University Benchmark (LUBM) [4] as a case where no disjunctive information is present and with the more expressive University Ontology Benchmark (UOBM) [8] using the HermiT[3] hypertableau reasoner. All experiments were performed on a Mac OS X Lion machine with a 2.53 GHz Intel Core i7 processor and Java 1.6 allowing 1GB of Java heap space. We measure the time for one-off tasks such as classification separately since such tasks are usually performed before the system accepts queries. The ontologies and all code required to perform the experiments are available online.[4]

We first used the 14 conjunctive ABox queries provided in LUBM. From these, queries 2, 7, 8, 9 are the most interesting ones in our setting since they contain many atoms and ordering them can have an effect in running time. We tested the queries on LUBM(1,0) and LUBM(2,0) which contain data for one or two universities respectively, starting from index 0. LUBM(1,0) contains 17,174 individuals and LUBM(2,0) contains 38,334 individuals. LUBM(1,0) took 19 s to load and 0.092 s for classification and initialization of known and possible instances of classes and properties. The clustering approach for classes took 1 s and resulted in 16 clusters. The clustering approach

[3] http://www.hermit-reasoner.com/

[4] http://code.google.com/p/query-ordering/

Table 2. Query answering times in milliseconds for LUBM(1,0) and LUBM(2,0) using i) the static algorithm ii) the dynamic algorithm, iii) 50% random sampling (RSampling), iv) the constructed individual clusters for sampling (CSampling)

	LUBM(1,0)				LUBM(2,0)			
Query	Static	Dynamic	RSampling	CSampling	Static	Dynamic	RSampling	CSampling
∗2	51	119	390	37	162	442	1,036	153
7	25	29	852	20	70	77	2,733	64
8	485	644	639	551	622	866	631	660
∗9	1,099	2,935	3,021	769	6,108	23,202	14,362	3,018

for properties lasted 4.9 s and resulted in 17 property successor clusters, 29 property predecessor clusters and 87 property clusters. LUBM(2,0) took 48.5 s to load and 0.136 s for classification and initialization of known and possible instances. The clustering approach for classes took 3.4 s and resulted in 16 clusters. The clustering approach for properties lasted 16.3 s and resulted in 17 property successor clusters, 31 property predecessor clusters and 102 property clusters. Table 2 shows the execution time for each of the four queries for LUBM(1,0) and LUBM(2,0) for four cases: i) when we use the static algorithm (columns 2 and 6), ii) when we use the dynamic algorithm (columns 3 and 7), iii) when we use random sampling, i.e., taking half of the individuals that are returned (from the evaluation of previous query atoms) in each run, to decide about the next cheapest atom to be evaluated in the dynamic case and iv) using the proposed sampling approach that is based on clusters constructed from individuals in the queried ontology (columns 4 and 8). The queries marked with (*) are the queries where the static and dynamic algorithms result in a different ordering. In queries 7 and 8 we observe an increase in running time when the dynamic technique is used (in comparison to the static) which is especially evident on query 8 of LUBM(2,0), where the number of individuals in the ontology and the intermediate result sizes are larger. Dynamic ordering also behaves worse than static in queries 2 and 9. This happens because, although the dynamic algorithm chooses a better ordering than the static algorithm, the intermediate results (that need to be checked in each iteration to determine the next query atom to be executed) are quite large and hence the cost of iterating over all possible mappings in the dynamic case far outweighs the better ordering that is obtained. We also observe that a random sampling for collecting the ordering statistics in the dynamic case (checking only 50% of individuals in Ω_{i-1} randomly for detecting the next query atom to be executed) leads to much worse results in most queries than plain static or dynamic ordering. This happens since random sampling often leads to the choice of a worse execution order. The use of the cluster based sampling method performs better than the plain dynamic algorithm in all queries. In queries 2 and 9, the gain we have from the better ordering of the dynamic algorithm is much more evident. This is the case since we use at most one individual from every cluster for the cost functions computation and the number of clusters is much smaller than the number of the otherwise tested individuals in each run.

In order to show the effectiveness of our proposed cost functions we compared the running times of all the valid plans (plans constructed according to Definition 3) with the running time of the plan chosen by our method. In the following we show the results

Table 3. Statistics about the constructed plans and chosen orderings and running times in milliseconds for the orderings chosen by Pellet and for the worst constructed plans

Query	PlansNo	Chosen Plan Order			Pellet Plan	Worst Plan
		Static	Dynamic	Sampling		
2	336	2	1	1	51	4,930
7	14	1	1	1	25	7,519
8	56	1	1	1	495	1,782
9	336	173	160	150	1,235	5,388

for LUBM(1, 0), but the results for LUBM(2,0) are comparable. In Table 3 we show, for each query, the number of plans that were constructed (column 2), the order of the plan chosen by the static, dynamic, and cluster based sampling methods if we order the valid plans by their execution time (columns 3,4,5; e.g., a value of 2 indicates that the ordering method chose the second best plan), the running time of HermiT for the plan that was created by Pellet[5] (column 6) as well as the running time of the worst constructed plan (column 7). The comparison of our ordering approach with the approach followed by other reasoners that support conjunctive query answering such as Pellet or Racer[6] is not very straightforward. This is the case because Pellet and Racer have many optimizations for instance retrieval [11,5], which HermiT does not have. Thus, a comparison between the execution times of these reasoners and HermiT would not convey much information about the effectiveness of the proposed query ordering techniques. The idea of comparing only the orderings computed by other reasoners with those computed by our methods is also not very informative since the orderings chosen by different reasoners depend much on the way that queries are evaluated and on the costs of specific tasks in these reasoners and, hence, are reasoner dependent, i.e., an ordering that is good for one reasoner and which leads to an efficient evaluation may not be good for another reasoner. We should note that when we were searching for orderings according to Pellet, we switched off the simplification optimization that Pellet implements regarding the exploitation of domain and range axioms of the queried ontology for reducing the number of query atoms to be evaluated [10]. This has been done in order to better evaluate the difference of the plain ordering obtained by Pellet and HermiT since our cost functions take into account all the query atoms.

We observe that for all queries apart from query 9 the orderings chosen by our algorithms are the (near)optimal ones. For queries 2 and 7, Pellet chooses the same ordering as our algorithms. For query 8, Pellet chooses an ordering which, when evaluated with HermiT, results in higher execution time. For query 9, our algorithms choose plans from about the middle of the order over all the valid plans w.r.t. query execution time, which means that our algorithms do not perform well in this query. This is because of the greedy techniques we have used to find the execution plan which take into account only local information to choose the next query atom to be executed. Interestingly, the use of cluster based sampling has led to the finding of a better ordering, as we can see from the running time in Table 2 and the better ordering of the plan found with cluster based

[5] http://clarkparsia.com/pellet/

[6] http://www.racer-systems.com

Table 4. Query answering times in seconds for UOBM (1 university, 3 departments) and statistics

Query	Static	Dynamic	CSampling	PlansNo	Chosen Plan Order			Pellet Plan	Worst Plan
					Static	Dynamic	Sampling		
4	13.35	13.40	13.41	14	1	1	1	13.40	271.56
9	186.30	188.58	185.40	8	1	1	1	636.91	636.91
11	0.98	0.84	1.67	30	1	1	1	0.98	> 30 min
12	0.01	0.01	0.01	4	1	1	1	0.01	> 30 min
14	94.61	90.60	93.40	14	2	1	1	> 30 min	> 30 min
q_1	191.07	98.24	100.25	6	2	1	1	> 30 min	> 30 min
q_2	47.04	22.20	22.51	6	2	1	1	22.2	> 30 min

sampling techniques compared to static or plain dynamic ordering (Table 3). The ordering chosen by Pellet for query 9 does also not perform well. We see that, in all queries, the worst running times are many orders of magnitude greater than the running times achieved by our ordering algorithms. In general, we observe that in LUBM static techniques are adequate and the use of dynamic ordering does not improve the execution time much compared to static ordering.

Unlike LUBM, the UOBM ontology contains disjunctions and the reasoner makes also nondeterministic derivations. In order to reduce the reasoning time, we removed the nominals and only used the first three departments containing 6,409 individuals. The resulting ontology took 16 s to load and 0.1 s to classify and initialize the known and possible instances. The clustering approach for classes took 1.6 s and resulted in 356 clusters. The clustering approach for properties lasted 6.3 s and resulted in 451 property successor clusters, 390 property predecessor clusters and 4,270 property clusters. We ran our static and dynamic algorithms on queries 4, 9, 11, 12 and 14 provided in UOBM, which are the most interesting ones because they consist of many atoms. Most of these queries contain one atom with possible instances. As we see from Table 4, static and dynamic ordering show similar performance in queries 4, 9, 11 and 12. Since the available statistics in this case are quite accurate, both methods find the optimal plans and the intermediate result set sizes are small. For both ordering methods, atoms with possible instances for these queries are executed last. In query 14, the dynamic algorithm finds a better ordering which results in improved performance. The effect that the cluster based sampling technique has on the running time is not as obvious as in the case of LUBM. This happens because in the current experiment the intermediate result sizes are not very large and, most importantly, because the gain obtained due to sampling is in the order of milliseconds whereas the total query answering times are in the order of seconds obscuring the small improvement in running time due to sampling. In all queries the orderings that are created by Pellet result in the same or worse running times than the orderings created by our algorithms.

In order to illustrate when dynamic ordering performs better than static, we also created the two custom queries:

$$q_1 = \{ \text{isAdvisedBy}(x,y), \text{GraduateStudent}(x), \text{Woman}(y) \}$$
$$q_2 = \{ \text{SportsFan}(x), \text{GraduateStudent}(x), \text{Woman}(x) \}$$

In both queries, P[GraduateStudent], P[Woman] and P[isAdvisedBy] are non-empty, i.e., the query concepts and properties have possible instances. The running times for dynamic ordering are smaller since the more accurate statistics result in a smaller number of possible instances that have to be checked during query execution. In particular, for the static ordering, 151 and 41 possible instances have to be checked in query q_1 and q_2, respectively, compared to only 77 and 23 for the dynamic ordering. Moreover, the intermediate results are generally smaller in dynamic ordering than in static leading to a significant reduction in the running time of the queries. Interestingly, query q_2 could not be answered within the time limit of 30 minutes when we transformed the three query classes into a conjunction, i.e., when we asked for instances of the intersection of the three classes. This is because for complex classes the reasoner can no longer use the information about known and possible instances and falls back to a more naive way of computing the class instances. Again, for the same reasons as before, the sampling techniques have no apparent effect on the running time of these queries.

6 Related Work

The problem of finding good orderings for the atoms of a query issued over an ontology has already been preliminarily studied [10,5].

Similarly to our work, Sirin et al. exploit reasoning techniques and information provided by reasoner models to create statistics about the cost and the result size of query atom evaluations within execution plans. A difference is that they use cached models for cheaply finding obvious class and property (non-)instances, whereas in our case we do not cache any model or model parts. Instead we process the pre-model constructed for the initial ontology consistency check and extract the known and possible instances of classes and properties from it. We subsequently use this information to create and update the query atom statistics. Moreover, Sirin et al. compare the costs of complete execution plans —after heuristically reducing the huge number of possible complete plans — and choose the one that is most promising before the beginning of query execution. This is different from our cheap greedy algorithm that finds, at each iteration, the next most promising query atom. Our experimental study shows that this is equally effective as the investigation of all possible execution orders. Moreover, in our work we have additionally used dynamic ordering combined with clustering techniques, apart from static ones, and have shown that these techniques lead to better performance particularly in ontologies that contain disjunctions and do now allow for purely deterministic reasoning.

Haarslev et al. discuss by means of an example the ordering criteria they use to find efficient query execution plans. In particular, they use traditional database cost based optimization techniques, which means that they take into account only the cardinality of class and property atoms to decide about the most promising ordering. As previously discussed, this can be inadequate especially for ontologies with disjunctive information.

7 Conclusions

In the current paper, we presented a method for ordering the atoms of a conjunctive instance query that is issued over an OWL ontology. We proposed a method for the definition of cost formulas that are based on information extracted from models of a reasoner (in our case HermiT). We have devised two algorithms, a static and a dynamic one, for finding a good order and show through an experimental study that static techniques are quite adequate for ontologies in which reasoning is deterministic. When reasoning is nondeterministic, however, dynamic techniques often perform better. The use of cluster based sampling techniques can improve the performance of the dynamic algorithm when the intermediate result sizes of queries are sufficiently large, whereas random sampling was not beneficial and often led to suboptimal query execution plans.

Future work will include the definition of additional cost measures and sampling criteria for ordering query atoms and the evaluation of our ordering techniques on a broader set of ontologies and queries.

Acknowledgements This work was done within the Transregional Collaborative Research Centre SFB/TRR 62 "Companion-Technology for Cognitive Technical Systems" funded by the German Research Foundation (DFG).

References

1. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P. (eds.): The Description Logic Handbook: Theory, Implementation, and Applications, 2nd edn. Cambridge University Press (2007)
2. Glimm, B., Horrocks, I., Lutz, C., Sattler, U.: Conjunctive query answering for the description logic SHIQ. Journal of Artificial Intelligence Research 31, 151–198 (2008)
3. Glimm, B., Horrocks, I., Motik, B., Shearer, R., Stoilos, G.: A novel approach to ontology classification. Journal of Web Semantics: Science, Services and Agents on the World Wide Web (accepted, 2012)
4. Guo, Y., Pan, Z., Heflin, J.: LUBM: A benchmark for OWL knowledge base systems. J. Web Semantics 3(2-3), 158–182 (2005)
5. Haarslev, V., Möller, R.: On the scalability of description logic instance retrieval. J. Autom. Reason. 41(2), 99–142 (2008)
6. Kazakov, Y.: $\mathcal{RIQ}$ and $\mathcal{SROIQ}$ are harder than $\mathcal{SHOIQ}$. In: Brewka, G., Lang, J. (eds.) Proc. 11th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR 2008), pp. 274–284. AAAI Press (2008)
7. 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)
8. Ma, L., Yang, Y., Qiu, Z., Xie, G., Pan, Y., Liu, S.: Towards a Complete OWL Ontology Benchmark. In: Sure, Y., Domingue, J. (eds.) ESWC 2006. LNCS, vol. 4011, pp. 125–139. Springer, Heidelberg (2006)
9. Motik, B., Shearer, R., Horrocks, I.: Hypertableau reasoning for description logics. Journal of Artificial Intelligence Research 36, 165–228 (2009)
10. Sirin, E., Parsia, B.: Optimizations for answering conjunctive ABox queries: First results. In: Proc. of the Int. Description Logics Workshop, DL (2006)
11. Sirin, E., Parsia, B., Grau, B.C., Kalyanpur, A., Katz, Y.: Pellet: A practical OWL-DL reasoner. J. of Web Semantics 5(2), 51–53 (2007)
12. Steinbrunn, M., Moerkotte, G., Kemper, A.: Heuristic and randomized optimization for the join ordering problem. VLDB Journal 6, 191–208 (1997)
13. Stocker, M., Seaborne, A., Bernstein, A., Kiefer, C., Reynolds, D.: SPARQL basic graph pattern optimization using selectivity estimation. In: Proceedings of the 17th International Conference on World Wide Web, WWW 2008, pp. 595–604. ACM, New York (2008)
14. Tsarkov, D., Horrocks, I., Patel-Schneider, P.F.: Optimizing terminological reasoning for expressive description logics. J. Autom. Reasoning 39(3), 277–316 (2007)

Robust Runtime Optimization and Skew-Resistant Execution of Analytical SPARQL Queries on Pig

Spyros Kotoulas[1,2,⋆], Jacopo Urbani[2,⋆], Peter Boncz[3,2], and Peter Mika[4]

[1] IBM Research, Ireland
Spyros.Kotoulas@ie.ibm.com
[2] Vrije Universiteit Amsterdam, The Netherlands
jacopo@cs.vu.nl
[3] CWI Amsterdam, The Netherlands
boncz@cwi.nl
[4] Yahoo! Research Barcelona, Spain
pmika@yahoo-inc.com

Abstract. We describe a system that incrementally translates SPARQL queries to Pig Latin and executes them on a Hadoop cluster. This system is designed to work efficiently on complex queries with many self-joins over huge datasets, avoiding job failures even in the case of joins with unexpected high-value skew. To be robust against cost estimation errors, our system *interleaves* query optimization with query execution, determining the next steps to take based on data samples and statistics gathered during the previous step. Furthermore, we have developed a novel skew-resistant join algorithm that replicates tuples corresponding to popular keys. We evaluate the effectiveness of our approach both on a synthetic benchmark known to generate complex queries (BSBM-BI) as well as on a Yahoo! case of data analysis using RDF data crawled from the web. Our results indicate that our system is indeed capable of processing huge datasets without pre-computed statistics while exhibiting good load-balancing properties.

1 Introduction

The amount of Linked Open Data (LOD) is strongly growing, both in the form of an ever expanding collection of RDF datasets available on the Web, as well semantic annotations increasingly appearing in HTML pages, in the form of RDFa, microformats, or microdata such as schema.org.

Search engines like Yahoo! crawl and process this data in order to provide more efficient search. Applications built on or enriched with LOD typically employ a warehousing approach, where data from multiple sources is brought together, and interlinked (e.g. as in [11]).

Due to the large volume and, often, dirty nature of the data, such Extraction, Transformation and Loading (ETL) processes can easily be translated into

⋆ These authors have contributed equally to this work.

"killer" SPARQL queries that overwhelm the current state-of-the-art in RDF database systems. Such problems typically come down to formulating joins that produce huge results, or to RDF database systems that calculate the wrong join order such that the intermediate results get too large to be processed. In this paper, we describe a system that scalably executes SPARQL queries using the Pig Latin language [16] and we demonstrate its usage on a synthetic benchmark and on crawled Web data. We evaluate our method using both a standard cluster and the large MapReduce infrastructure provided by Yahoo!.

In the context of this work, we are considering some key issues:

Schema-Less. A SPARQL query optimizer typically lacks all schema knowledge that a relational system has available, making this task more challenging. In a relational system, schema knowledge can be exploited by keeping statistics, such as table cardinalities and histograms that capture the value and frequency distributions of relational columns. RDF database systems, on the other hand, cannot assume any schema and store all RDF triples in a table with Subject, Property, Object columns (S,P,O). Both relational column projections as well as foreign key joins map in the SPARQL equivalent into self-join patterns. Therefore, if a query is expressed in both SQL and SPARQL, on the physical algebra level, the plan generated from SPARQL will typically have many more self-joins than the relational plan has joins. Because of the high complexity of join order optimization as a function of the number of joins, SPARQL query optimization is more challenging than SQL query optimization.

MapReduce and Skew. Linked Open Data ETL tasks which involve cleaning, interlinking and inferencing have a high computational cost, which motivates our choice for a MapReduce approach. In a MapReduce-based system, data is represented in files and that can come from recent Web crawls. Hence, we have an initial situation *without* statistics and *without* any B-trees, let alone multiple B-trees. One particular problem in raw Web data is the high skew in join keys in RDF data. Certain subjects and properties are often re-used (most notorious are RDF Schema properties) which lead to joins where certain key-values will be very frequent. These keys do not only lead to large intermediate results, but can also cause one machine to get overloaded in a join job and hence run out of memory (and automatic job restarts provided by Hadoop will fail again). This is indeed more general than joins: in the sort phase of MapReduce, large amounts of data might need to be sorted on disk, severely degrading performance.

SPARQL on Pig. The Pig Latin language provides operators to scan data files on a Hadoop cluster that form tuple streams, and further select, aggregate, join and sort such streams, both using ready-to-go Pig relational operators as well as using user-defined functions (UDFs). Each MapReduce job materializes the intermediate results in files that are replicated in the distributed filesystem. Such materialization and replication make the system robust, such that the jobs of failing machines can be restarted on other machines without causing a query failure. However, from the query processing point of view, this materialization is a source of inefficiency. The Pig framework attempts to improve this situation by compiling Pig queries into a minimal amount of Hadoop jobs, effectively combining more operators in a single MapReduce operation. An efficient query optimization strategy must be aware of it and each query processing step should minimize the number of Hadoop jobs.

Our work addresses these challenges and proposes an efficient translation of some crucial operators into Pig Latin, namely joins, making them robust enough to deal with the issues typical of large data collections.

Contributions. We can summarize the contributions of this work as follows. *(i)* We have created a system that can compute complex SPARQL queries on huge RDF datasets. *(ii)* We present a runtime query optimization framework that is optimized for Pig in that it aims at reducing the number of MapReduce jobs, therefore reducing query latency. *(iii)* We describe a skew-resistant join method that can be used when the runtime query optimization discovers the risk for a skewed join distribution that may lead to structural machine overlap in the MapReduce cluster. *(iv)* We evaluate the system on a standard cluster and a Yahoo! Hadoop cluster of over 3500 machines using synthetic benchmark data, as well as real Web crawl data.

Outline. The rest of the paper is structured as follows. In Section 2 we present our approach and describe some of its crucial parts. In Section 3 we evaluate our approach on both synthetic and real-world data. In Section 4 we report on related work. Finally, in Section 5, we draw conclusions and discuss future work.

2 SPARQL with Pig: Overview

In this section, we present a set of techniques to allow efficient querying over data on Web-scale, using MapReduce. We have chosen to translate the SPARQL 1.1 algebra to Pig Latin instead of making a direct translation to a physical algebra in order to readily exploit optimizations in the Pig engine. While this work is the first attempt to encode full SPARQL 1.1 in Pig, a complete description of such process is elaborate and goes beyond the scope of this paper.

The remaining of this section is organized as follows: in Sections 2.1 and 2.2, we present a method for runtime query optimization and query cost calculation suitable for a batch processing environment like MapReduce. Finally, in subsection 2.3, we present a skew detection method and a specialized join predicate suitable for parallel joins under heavy skew, frequent on Linked Data corpora.

2.1 Runtime Query Optimization

We adapt the ROX query optimization algorithm [1,10] to SPARQL and MapReduce. ROX interleaves query execution with join sampling, in order to improve result set estimates. Our specific context differs to that of ROX in that:

– SPARQL queries generate a large number of joins, which often have a multi-star shape [5].

Algorithm 1. Runtime optimization

```
1  $J$: Set of join operators in the query
2  $L$: List of sets of (partial) query plans
3  void optimize joins($J$) {
4   execute(J)
5   $L_0$:=(J)
6   $i$:=1
7   while ($L_{i-1} \neq \emptyset$)
8    for (j $\in L_{i-1}$)
9     for ($I \in \{L_0 ... L_{i-1}\}$)
10     for (k $\in I$)
11      if (j $\neq$ k)
12       $L_i$.add(construct(j,k))
13    if (stddev(cost($L_i$))/mean(cost($L_i$)) $>$ t)
14     prune($L_i$)
15     sample($L_i$)
16    $i$:=$i+1$
17 }
```

– The overhead of starting MapReduce jobs in order to perform sampling is significant. The start-up latency for *any* MapReduce job lies within tens of seconds and minutes.
– Given the highly parallel nature of the environment, executing several queries at the same time has little impact on the execution time of each query.

Algorithm 1 outlines the basic block of our join order optimization algorithm. To cope with the potentially large number of join predicates in SPARQL queries, we draw from dynamic programming and dynamic query optimization techniques, constructing the plans bottom-up and partially executing them.

Initially, we extract from the dataset the binding for all the statement patterns in the query and calculate their cardinalities. From initial experiments, given the highly parallel nature of MapReduce, we have concluded that the cost of this operation is amortized over the execution of the query since we are avoiding several scans over the entire input. Then, we perform a series of *construct-prune-sample* cycles. The construct phase generates new solutions from the partial solutions in the previous cycles. These are then pruned according to their estimated cost. The remaining ones are sampled and/or partially executed. The pruning and sampling phases are optional. We will only sample if $stddev(costs)/mean(costs)$ is higher than some given threshold, so as to avoid additional optimization overhead if the cost estimates for the candidate plans are not significantly different.

Construct. During the construct phase (lines 6-10 in Algorithm 1), the results of the previous phases are combined to generate new candidate (partial) plans. A new plan is generated by either adding an argument to an existing node when possible (e.g. making a 2-way join a 3-way join) or by creating a new join node.

Algorithm 2. Bi-focal sampling in Pig

```
DEFINE bifocal_sampling(L, R, s, t)
 RETURNS FC {
LS = SAMPLE L s;
RS = SAMPLE R s;
LSG = GROUP LS BY S;
RSG = GROUP RS BY S;
LSC = FOREACH LSG GENERATE flatten(group), COUNT(LSG) as c;
RSC = FOREACH RSG GENERATE flatten(group), COUNT(RSG) as c;
LSC = FOREACH LSC GENERATE group::S as S ,c as c;
RSC = FOREACH RSC GENERATE group::S as S ,c as c;
SPLIT LSC INTO LSCP IF c>=t, LSCNP IF c<t;
SPLIT RSC INTO RSCP IF c>=t, RSCNP IF c<t;
// Dense
DJ = JOIN LSCP BY S, RSCP BY S using 'replicated';
DJ = FOREACH RA GENERATE LSCP::c as c1, RSCP::c as c2;
// Left sparse
RA = JOIN RSC BY S, LSCNP BY S;
RA = FOREACH RA GENERATE LSCNP::c as c1, RSC::c as c2;
// Right sparse
LA = JOIN LSC BY S, RSCNP BY S;
LA = FOREACH LA GENERATE LSC::c as c1, RSCNP::c as c2;
// Union results
AC = UNION ONSCHEMA DA, RA, LA;
$FC = FOREACH AC GENERATE c1*c2 as c;}
```

Prune. We pick the $k\%$ cheapest plans in the previous phase, using the cost calculation mechanism described in Section 2.2. The remaining plans are discarded.

Sample. To improve the accuracy of the estimation, we fully execute the plan up to depth 1 (i.e. the entire plan minus the upper-most join). Then, we use Algorithm 2 to perform bi-focal sampling [6].

There is a number of salient features in our join optimization algorithm:

– There is a degree of speculation, since we are sampling only after constructing and pruning the plans. We do not select plans based on their calculated cost using sampling, but we are selecting plans based on the cost of their 'subplans' and the operator that will be applied.
– Nevertheless, our algorithm will not get 'trapped' into an expensive join, since we only fully execute a join after we have sampled it in a previous cycle.
– Since we are evaluating multiple partial solutions at the same time, it is essential to re-use existing results for our cost estimations and to avoid unnecessary computation. Since the execution of Pig scripts and our run-time optimization algorithm often materialize intermediate results anyway, the latter are re-used whenever possible.

2.2 Pig-Aware Cost Estimation

Using a MapReduce-based infrastructure gives rise to new challenges in cost estimation. First, queries are executed in batch and there is significant overhead in starting new batches. Second, within batches, there is no opportunity for sideways information passing [14], due to constraints in the programming model. Third, when executing queries on thousands of cores, load-balancing becomes very important, often outweighing the cost for calculating intermediate results. Fourth, random access to data is either not available or very slow since there are no data indexes. On the other hand, reading large portions of the input is relatively cheap, since it is an embarrassingly parallel operation.

In this context, we have developed a model based on the cost of the following: Writing a tuple (w); Reading a tuple (r); The cost of a join per tuple. In Hadoop, a join can be performed either during the reduce phase (j_r), essentially a combination of a hash-join between machines and a merge-join on each machine, or during the map phase (j_m), by loading one side in the memory of all nodes, essentially a hash-join. Obviously, the latter is much faster than the former, since it does not require repartitioning of the data on one side or sorting, exhibits good load-balancing properties, and requires that the input is read and written only once; The depth of the join tree(d), when considering only the reduce-phase joins. This is roughly proportional to the number of MapReduce jobs required to execute the plan. Considering the significant overhead of executing a job, we consider this separately from reading and writing tuples.

The final cost for a query plan is calculated as the weighted sum of the above, with indicatory weights being 3 for w, 1 for r, 10 for j_r, 1 for j_m and a value proportional to the size of the input for d.

2.3 Dealing with Skew

The significant skew in the term distribution of RDF data has been recognized as a major barrier to efficient parallelization [12]. In this section, we are presenting a method to detect skew and a method for load-balanced joins in Pig.

Detecting Skew. To detect skew (and estimate result set size), we are presenting an implementation of bi-focal sampling [6] for Pig and report the pseudocode in Algorithm 2. Similar to join optimization, one of the main goals is to minimize the number of jobs. L, R, s and t refer to the left side of the join, the right side of the join, the sampling rate and the number of tuples that the memory can hold respectively. Initially, we sample the input (lines 3-4), group by the join keys (lines 6-7) and count the number of occurrences of each key (lines 7-10). We split each side of the join by key popularity using a fixed threshold, which is dependent on the amount or memory available to each processing node (lines 11-12). We then perform a join between tuples with popular keys (lines 14-15) and a join for each side for tuples with non-popular keys and the entire input (lines 17-21).

This algorithm generates seven MapReduce jobs out of which two are Map-only jobs that can be executed in parallel and four are jobs with Reduce phases that happen concurrently in pairs. In fact, it is possible to implement our algorithm in two jobs, programming directly on Hadoop instead of using Pig primitives.

Determining Join Implementation. In Pig, it is up to the developer to choose the join implementation. In our system, we choose according to the following:

– If all join arguments but one fit in memory, then we perform a replicated join. Replicated joins are performed on the Map side by loading all arguments except for one into main memory and streaming through the remaining one.
– If we have a join with more than two arguments and more than one of them does not fit in memory, we are performing a standard (hash) join.
– If the input arguments or the results of the (sampled) join present significant skew, we perform the skew-resistant join described in the following section.

Skew-Resistant Join. As a by-product of the bi-focal sampling technique presented previously, we have the term distribution for each side of the join and an estimate of the result size for each term. Using this information, we can estimate the skew as the ratio of the maximum number of results for any key to the average number of results over all keys. Hadoop has some built-in resistance to skewed joins by means of rescheduling jobs to idle nodes, which is sufficient for cases where some jobs are slightly slower than others. Furthermore, Pig has a specialized join predicate to handle a skewed join key popularity [7], by virtue of calculating a key popularity histogram and distributing the jobs according to this. Nevertheless, neither of these algorithms can effectively handle skewed joins where a very small number of keys dominates the join. We should further note that, since there is no communication between nodes after a job execution has started, a skewed key distribution will cause performance problems even if the hit rate for those keys is low. This is because MapReduce will still need to send all the tuples corresponding to these popular keys to a single reduce task.

Our algorithm executes a replicated join for the keys that have a highly skewed distribution in the input and a standard join for the rest. In other words, joining on keys that are responsible for load unbalancing is done by replicating one side on each machine and performing a local hash join. For the remaining keys, the join is executed by grouping the two sides by the join key and assigning the execution of each group to a different machine (as is standard in Pig). In Algorithm 3, we present the Pig Latin code for an example join of expressions A and B[1], on positions O and S respectively. Initially, we sample and extract the top-k popular terms for each side (lines 3-12), PopularA and PopularB respectively. Then, for each side of the input, we perform two left joins to associate tuples with PopularA and PopularB (lines 14-17). This allows us to split each of our inputs to three sets (lines 19-25), marked accordingly in Figure 1:

1. The tuples that correspond to keys that are popular on the other side (e.g. for expression A the keys in PopularB). For side B, we put an additional

[1] For brevity, we have omitted some statements that project out columns that are not relevant for our algorithm.

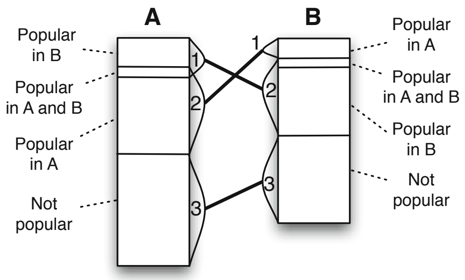


Fig. 1. Schematic representation of the joins to implement the skew-resistant join

requirement, namely that the key is *not* in PopularB. This is done to avoid producing the results for tuples that are popular twice.
2. The tuples that correspond to keys that are popular on the same side (e.g. for expression A the keys in PopularA).
3. The tuples that do not correspond to any popular keys on either side.

We use the above to perform replicated joins for the tuples corresponding to popular terms and standard joins for the tuples that are not. The tuples in A corresponding to popular keys in A (APopInA) are joined with the tuples in B that correspond to popular keys in A using a replicated join (line 28). The situation is symmetric for B (line 29). The tuples that do not correspond to popular keys from either side are processed using a standard join (line 31). The output of the algorithm is the union of the results of the three joins.

We should note that our algorithm will fail if APopInB and BPopInA are not small enough to be replicated to all nodes. But this can only be true if there are some keys that are popular in both sets. Joins with such keys would anyway lead to an explosion in the result set (since the result size of each of these popular keys is the product of their appearances in each side).

3 Evaluation

We present an evaluation of the techniques presented in this paper using synthetic and real data, and compare our approach to a commercial RDF store.

We have used two different Hadoop clusters in our evaluation: a modest cluster, part of the DAS-4 distributed supercomputer, and a large cluster installed at Yahoo!. The former was used to perform experiments in isolation and consists of 32 dual-core nodes, with 4GB of RAM and dual HDDs in RAID-0. The Yahoo! Hadoop cluster we have used in our experiment consists of over 3500 nodes, each with two quad-core CPUs, 16 GB RAM and 900GB of local space. This cluster is used in a utility computing fashion and thus we do not have exclusive access, meaning that we can not exploit the full capacity of the cluster and our runtimes at any point might be (negatively) influenced by the jobs of other users. We thus only report actual, but not best possible performance.

In order to compare our approach with existing solutions, we deployed Virtuoso v7 [4], a top-performing RDF store, on an extremely high-end server: a

Algorithm 3. Skew-resistant join

```
DEFINE skew_resistant_join(A, B, k)
 RETURNS result {
SA = SAMPLE A 0.01; //Sample first side
GA = GROUP SA BY O;
GA2 = FOREACH GA GENERATE COUNT_STAR(SA), group;
OrderedA = ORDER GA2 BY $0 DESC;
PopularA = LIMIT OrderedA k;
SB = SAMPLE B 0.01; //Sample second side
GB = GROUP SB BY S;
GB2 = FOREACH GB GENERATE COUNT_STAR(SB), group;
OrderedB = ORDER GB2 BY $0 DESC;
PopularB = LIMIT OrderedB k;

PA = JOIN A BY O LEFT, PopularA BY O USING 'replicated';
PPA= JOIN PA BY O LEFT, PopularB BY S USING'replicated';
PB = JOIN B BY S LEFT, PopularB BY S USING 'replicated';
PPB= JOIN PB BY S LEFT, PopularA BY S USING 'replicated';

SPLIT PPA INTO APopInA IF PopularA::O is not null,
   APopInB IF PopularB::S is not null, ANonPop IF
   PopularA::0 is not null and PopularB::S is not null;

SPLIT PPB INTO BPopInB IF PopularB::S is not null, BPopInA
   IF PopularA::O is not null and PopularB::S is null, BNonPop
   IF PopularA::0 in not null and PopularB::S is not null;

 // Perform replicated joins for popular keys
 JA= JOIN BPopInB BY S, APopInB BY O USING 'replicated';
 JB= JOIN APopInA BY S, BPopInA BY S USING 'replicated';
// Standard join for non-popular keys
 JP = JOIN ANonPop BY O, BNonPop BY S;

$result = UNION ONSCHEMA JA, JB, JP; }
```

4-socket 2.4GHz Xeon 5870 server (40 cores) with 1TB of main memory and 16 magnetic disks in RAID5, running Red Hat Enterprise Linux 6.0.

We chose two datasets for the evaluation. Firstly, the Berlin SPARQL benchmark [3], Business Intelligence use-case v3.1 (BSBM-BI). This benchmark consists of 8 analytical query patterns from the e-commerce domain. The choice for this benchmark is based on the scope of this work, namely complex SPARQL queries from an analytical RDF workload.

Secondly, we also used our engine for some analytical queries on RDF data that Yahoo! has crawled from the Web. This data is a collection of publicly available information on the Web encoded or translated in RDF. The dataset

Query	1B DAS4	1B Y!	10B Y!
1	38m	1h50m	1h17m
2	13m	23m	31m
3	17m	18m	24m
4	38m	1h34m	54m
5	1h4m	3h10m	1h52m
6	48m	34m	1h19m
7	26m	43m	46m
8	1h	1h59m	1h38m

(a) Execution time of the BSBM queries on 1B data on the DAS-4 and Yahoo! cluster

Query	Cold runtime	Warm runtime
1	3m46s	1m48s
2	41s	15s
3	29m	24m5s
4	52m6s	50m55s
5	11m42s	5m19s
6	6s	0.06s
7	50s	9ms
8	39m58s	36m22s

(b) Runtime of the BSBM queries using Virtuoso

Fig. 2. BSBM query execution time

that we used consists of about 26 billion triples that correspond to about 3.5 terabytes of raw data (328 gigabytes compressed).

3.1 Experiments

In our evaluation, we measured: (i) *the performance of our approach for large datasets*. To this end, we launched and recorded the execution time of all the queries on BSBM datasets of 1 and 10 billion triples. (ii) *The effectiveness of our dynamic optimization technique*. To this purpose, we measured the cost of this process and what is its effect on the overall performance. (iii) *The load-balancing properties of our system*. To this end, we have performed a high-level evaluation of the entire querying process in terms of load balancing, and we have further focused on the performance of the skew-resistant join, which explicitly addresses load-balancing issues.

General Performance. We have launched all 8 BSBM queries on 1 billion triples on both clusters and on 10 billion triples using only the Yahoo! cluster.

In Figure 2a, we report the obtained runtimes. We make the following observations: First of all, the fastest queries (queries 2 and 3) have a runtime of a bit less than 20 minutes on the DAS-4 cluster. The slowest is query 5, with a runtime of about one hour. For the 1B-triple dataset, the execution times on the Yahoo! cluster are significantly higher than those on the DAS-4 cluster. This is due to (a) the fact that the Yahoo! cluster is shared with other users, so, we often need to wait for the jobs of other users to finish execution and (b) the much larger size of the Yahoo! cluster, introducing additional coordination overhead.

We also note that the runtime does not proportionally increase with the data size on the Yahoo! cluster: the runtimes for the one and ten billion-triple datasets are comparable. Such behavior is explained by the fact that a proportional amount of resources is allocated to the size of the input and the (significant) overhead to start MapReduce jobs does not increase. Our approach, combined

Listing 1.1. Exploratory SPARQL queries

```
SELECT (count(?s) as ?f) (min(?s) as ?ex) ?ct ?di ?mx ?mi{
    {SELECT ?s (count(?s) as ?ct) (count(distinct ?p) as ?di) (max(?p) as ?mx)
        (min(?p) as ?mi) {?s ?p ?o.} GROUP BY ?s}
    GROUP BY ?ct ?di ?mx ?mi ORDER BY desc(?f) LIMIT 10000}

SELECT ?p (COUNT(?s) AS ?c) {?s ?p ?o.} GROUP BY ?p ORDER BY ?c

SELECT ?C (COUNT(?s) AS ?n ) {?s a ?C.} GROUP BY ?C ORDER BY ?n
```

with the large infrastructure at Yahoo! allows us to scale to much larger inputs while keeping the runtime fairly constant.

To verify the performance in a real-world scenario and on messy data, we have launched three non-selective SPARQL queries, reported in Listing 1.1, over an RDF web crawl of Yahoo!. The first query is used for identifying "characteristic sets" [13]: frequently co-occurring properties with a subject. The second identifies all the properties used in the dataset and sorts them according to their frequency. The third identifies the classes with the most instances. These queries are typical of an exploratory ETL workload on messy data, aimed at creating a basic understanding of the structure and properties of a web-crawled dataset.

From the computation point of view, the first two queries have non-bound properties and the last one has a very non-selective property (*rdf:type*) . Therefore they will touch the entire dataset, including problematic properties that cause skew problems. The runtime of these three queries was respectively 1 hour and 21 minutes, 29 minutes and 25 minutes. The first query required 6 MapReduce jobs to be computed. The second and third each required 3 jobs.

It is interesting to compare the performance for BSBM against a standard RDF store (Virtuoso), even if the approaches are radically different. We loaded 10 billion BSBM triples on the platform described previously. This process took 61 hours (about 2.5 days) and was performed in collaboration with the Virtuoso development team to ensure that the configuration was optimal.

We executed the 8 BSBM queries used in this evaluation and we report the results in Figure 2b. For some queries, Virtuoso is many orders of magnitude faster than our approach (namely, for the simpler queries like queries 1,2, 6 and 7). For the more expensive queries, the difference is less pronounced. However, this performance comes at the price of a loading time of **61 hours**, necessary to create the database indexes. To load the data and run all the queries on Virtuoso, the total runtime amounts to 63 hours, while in our MapReduce approach, it amounts to 8 hours and 40 minutes. Although we can not generalize this conclusion to other datasets, the loading cost of Virtuoso is not amortized for a single query mix in BSBM-BI.

In this light, the respective advantages of the two systems are in running many cheap queries for Virtuoso and running a limited number of expensive queries for our system. Furthermore, our system can exploit existing Hadoop installations and run concurrently with heterogenous workloads.

Table 1. Breakdown of query runtimes on 1B BSBM data

Query	Cost input extraction	Cost dyn. optimizer	Final query execution
1	5m58s	12m12s	19m43s
2	4m22s	n.a.	9m5s
3	5m46s	n.a.	11m58s
4	6m22s	14m41s	16m55s
5	7m37s	34m3s	23m21s
6	8m12s	14m25s	25m10s
7	5m17s	6m17s	14m3s
8	8m7s	25m14s	27m2s

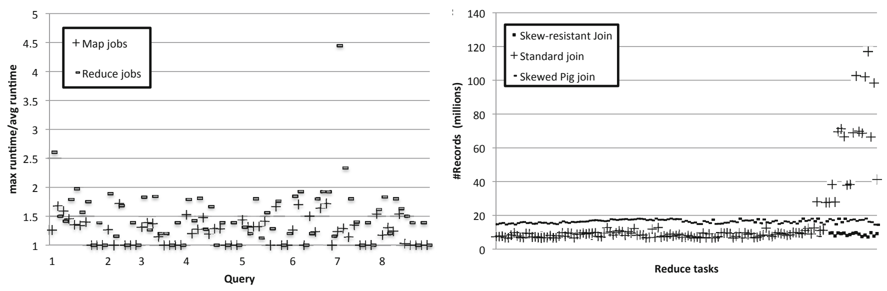


Fig. 3. Maximum task runtime divided by the average task runtime for a query mix (left). Comparison of load distribution between the skew-resistant join and the standard Pig join (right).

Dynamic Optimizer. As discussed in Section 2.1, query execution consists of three phases: (a) triple pattern extraction, (b) best execution plan identification using dynamic query optimization and (c) full query plan execution.

In Table 1, we summarize the execution time of each of these three phases for the 8 BSBM queries on the DAS-4 cluster, using the one billion triple dataset. We observe that extracting the input patterns is an operation that takes between 4 and 8 minutes, depending on the size of the patterns. Furthermore, the cost of dynamic optimization is significant compared to the total query execution time and largely depends on the complexity of the query, although, in our approach, part of the results are calculated during the optimization phase.

Load Balancing. As the number of processing nodes increases, load-balancing becomes increasingly important. In this section, we present the results concerning the load balancing properties for our approach.

Due to the synchronization model of MapReduce, it is highly desirable that no task takes significantly longer than others. In Figure 3 (left), we present the maximum task execution time, divided by the average execution time for all tasks launched for a query mix on the DAS-4 cluster. The x axis corresponds both to time and the queries that correspond to each job, the y axis corresponds to the time it took to execute the slowest task divided by the time it took to execute a task, on average. In a perfectly load-balanced system, the values in y would be equal to 1. In our system, except for a single outlier, the slowest tasks generally take less than twice the average time, indicating that the load balancing of our system is good. A second observation is that the load imbalance is higher in the reduce jobs. This is expected, considering that, in our system, the map tasks are typically concerned with partitioning the data (and process data chunks of equal size), while the reduce tasks process individual partitions.

Data skew becomes increasingly problematic as the number of processing nodes increases, since it generates unbalance between the workload of each node [12]. In the set of experiments described in this section, we analyze the performance of the skew-resistent join that we have introduced in Section 2.3 to efficiently execute joins on data with high skew.

To this purpose, we launched an expensive join using the 1 billion BSBM dataset and we analyzed the performance of the standard and the skew-resistant join. Our experiments were performed on the DAS-4 cluster, since we required a dedicated cluster to perform a comparative analysis. Considering that this platform uses only 32 nodes in total, the effect on the Yahoo! cluster would have been much more pronounced (since it is several orders of magnitude bigger).

We launched a join that used the predicate of the triples as a key; namely, we have performed a join of pattern *(?s ?p ?o)* with pattern *(?p ?p1?o1)*. Such joins are common in graph analysis, dataset characterization [13] and reasoning workloads (e.g. RDFS rules 2-3 and 7).

The runtime using the classical join was of about 1 hour and 29 minutes. On the other side, the runtime using the skew-resistant join was about 57 minutes. Therefore, such a join has a significant impact on performance, in the presence of skew. The impact is even higher if we consider that the skew-resistant join requires 21 jobs to finish while the classical job requires a single one.

The reason behind such increase of performance lies in the way the join is performed, and in particular, the amount and distribution of work that the reduce tasks need to do, as reported in Figure 3 (right). We see that some reducers receive a much larger number of records than others (these are the ones at the end of the x axis), implying that some nodes will need to perform much more computation than others. With the skew-resistant join, all the joins among popular terms are performed in the map phase and as a result, all the reducers receive a similar amount of tuples in the input.

We also report reduce task statistics for the the skew-resistant join implementation in Pig, which calculates a histogram for join keys to better distribute them across the join tasks. Although the size of the cluster is small enough to ensure an even load-balance, we note that the standard Pig skewed join sends almost double the number of records to each reduce task. This is attributed to fact that our approach shifts much of the load for joining to the Map phase.

4 Related Work

We have compared our approach with previous results from three related areas: (i) MapReduce query processing (ii) adaptive and sampling-based query optimization and (iii) cluster-aware SPARQL systems.

In the relational context, similar efforts towards SQL query processing over a MapReduce cluster are e.g. Hive [21] and HadoopDB [2]. Both projects do not provide query optimization when data is raw and unprocessed. Data import is a necessary first step in HadoopDB and may be costly. In Hive, query optimization based on statistics is only available if the data has been analyzed as a prior step.

On-the-fly query optimization in Manimal [9] analyzes MapReduce jobs on-the-fly and tries to enhance them by inserting compression code and sometimes even on-the-fly indexing and index lookup steps.

Situations where there is absence of data statistics in the relational context of query optimization has led to work on sampling and run-time methods. Our work reuses the bi-Focal sampling algorithm [6] which came out of the work in the relational community to use sampling for query result size estimation. In this work, we have adapted the bi-focal algorithm using the Pig language.

The rigid structure of MapReduce and high latencies in starting new jobs led us to adjust the dynamic re-optimization strategies to these constraints. Other interesting run-time approaches are sideways-information passing [14] in large RDF joins. These are not easily adaptable to the constraints of MapReduce.

With the ever growing sizes of RDF data available, scalability has been a primary concern and major RDF systems such as Virtuoso [4], 4store [18], and BigData [20] have evolved to parallel database architectures targeting cluster hardware. RDF systems typically employ heavy indexing, going as far as creating replicated storage in all six permutations of triple order [15,22], which makes data import a heavy process. Such choice puts them in a disadvantage when the scenario involves processing huge amounts of raw data. As an alternative to the parallel database approach, there are several other projects that process SPARQL queries over MapReduce. PIGSparQL [19] performs a direct mapping of SPARQL to Pig without focusing on optimization. RAPID+ [17] provides a limited form of on-the-fly optimization where *look-ahead processing* tries to combine multiple subsequent join steps. The adaptiveness of this approach is however limited compared to our sampling based run-time query optimization.

5 Conclusions

In this paper, we have presented an engine for the processing of complex analytic SPARQL 1.1 queries, based on Apache Pig. In particular, we have developed: (i) a translation from SPARQL 1.1 to Pig Latin, (ii) a method for runtime join optimization based on a cost model suitable for MapReduce-based systems, (iii) a method for result set estimation and join key skew detection, and (iv) a method for skew-resistant joins written in Pig. We have evaluated our approach on a standard and a very large Hadoop cluster used at Yahoo! using synthetic benchmark data and real-world data crawled from the Web.

In our evaluation, we established that our approach can answer complex analytical queries over very large synthetic data (10 Billion triples from BSBM) and over the largest real-world messy dataset in the literature (26 Billion triples). We compared our performance against a state-of-the-art RDF store on a large-memory server, even though the two approaches bear significant differences. While our approach is not competitive in terms of query response time, our system has the advantage that it does not require a-priori loading of the data, and thus has far better loading plus querying performance. Furthermore, our system runs on a shared architecture of thousands of machines, significantly easing deployment and potentially scaling to even larger volumes of data. We verified that the load in our system is well-balanced and our skew-resistant join significantly outperforms the standard join of Pig for skewed key distributions in the input.

In this work, for the first time, it has been shown that MapReduce is suited for very large-scale analytical processing of RDF graphs and it is, in fact, better suited than a traditional RDF store in a setting where a relatively small number of queries will be posted on a very large dataset.

We see future work in optimizing our architecture to further reduce overhead. This could be achieved by turning to an approach that adaptively indexes part of the input or performs part of the computation outside of Pig so as to reduce the number of jobs. Similarly, parts of the skew-resistant join can be already calculated during Bi-focal sampling (e.g. sampling and extracting the popular terms for the input relations).

Although in this paper we have presented our algorithm to handle skewed joins in the context of Pig, we expect that the result is transferrable to a general parallel data-processing framework.

Summarizing, in this paper, we have presented a technique with which technologies like MapReduce and Pig can be employed for large-scale SPARQL querying. The presented results are promising and set the lead for a new viable alternative to traditional RDF stores for executing expensive analytical queries on large volumes of RDF data.

Acknowledgements. Spyros Kotoulas and Jacopo Urbani were partially supported by EU project LarKC (FP7-215535) and the Yahoo! Faculty Research Engagement Program. Peter Boncz was partially supported by EU project LOD2 (FP7-257943).

References

1. Abdel Kader, R., Boncz, P., Manegold, S., van Keulen, M.: ROX: run-time optimization of XQueries. SIGMOD (2009)
2. Abouzeid, A., Bajda-Pawlikowski, K., Abadi, D., Rasin, A., Silberschatz, A.: HadoopDB: An Architectural Hybrid of MapReduce and DBMS Technologies for Analytical Workloads. PVLDB 2(1), 922–933 (2009)
3. Bizer, C., Schultz, A.: The Berlin SPARQL Benchmark. International Journal on Semantic Web and Information Systems (IJSWIS) 5(2), 1–24 (2009)
4. Erling, O.: Virtuoso, a Hybrid RDBMS/Graph Column Store. DEBULL 35(1), 3–8 (2012)

5. Gallego, M., Fernández, J., Martínez-Prieto, M., Fuente, P.: An empirical study of real-world SPARQL queries. In: USEWOD 2011 at WWW (2011)
6. Ganguly, S., Gibbons, P., Matias, Y., Silberschatz, A.: Bifocal sampling for skew-resistant join size estimation. SIGMOD Record 25(2), 271–281 (1996)
7. Gates, A., Natkovich, O., Chopra, S., Kamath, P., Narayanamurthy, S., Olston, C., Reed, B., Srinivasan, S., Srivastava, U.: Building a high-level dataflow system on top of Map-Reduce: the Pig experience. PVLDB 2(2), 1414–1425 (2009)
8. Ivanova, M., Kersten, M., Nes, N., Gonçalves, R.: An architecture for recycling intermediates in a column-store. TODS 35(4), 24 (2010)
9. Jahani, E., Cafarella, M., Ré, C.: Automatic Optimization for MapReduce Programs. PVLDB 4(6), 385–396 (2011)
10. Kader, R., van Keulen, M., Boncz, P., Manegold, S.: Run-time Optimization for Pipelined Systems. In: Proceedings of the IV Alberto Mendelzon Workshop on Foundations of Data Management (2010)
11. 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)
12. Kotoulas, S., Oren, E., van Harmelen, F.: Mind the data skew: distributed inferencing by speeddating in elastic regions. In: WWW (2010)
13. Neumann, T., Moerkotte, G.: Characteristic sets: Accurate cardinality estimation for RDF queries with multiple joins. In: ICDE (2011)
14. Neumann, T., Weikum, G.: Scalable join processing on very large RDF graphs. SIGMOD (2009)
15. Neumann, T., Weikum, G.: The RDF-3X engine for scalable management of RDF data. The VLDB Journal 19(1), 91–113 (2010)
16. Olston, C., Reed, B., Srivastava, U., Kumar, R., Tomkins, A.: Pig latin: a not-so-foreign language for data processing. SIGMOD (2008)
17. Ravindra, P., Deshpande, V., Anyanwu, K.: Towards scalable RDF graph analytics on MapReduce. In: Proceedings of the 2010 Workshop on Massive Data Analytics on the Cloud, p. 5. ACM (2010)
18. Salvadores, M., Correndo, G., Harris, S., Gibbins, N., Shadbolt, N.: The Design and Implementation of Minimal RDFS Backward Reasoning in 4store. 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. 139–153. Springer, Heidelberg (2011)
19. Schätzle, A., Lausen, G.: PigSPARQL: mapping SPARQL to Pig Latin. In: SWIM: The 3th International Workshop on Semantic Web Information Management (2011)
20. L. SYSTAP. Bigdata®
21. Thusoo, A., Sarma, J., Jain, N., Shao, Z., Chakka, P., Zhang, N., Anthony, S., Liu, H., Murthy, R.: Hive: a petabyte scale data warehouse using Hadoop. In: ICDE (2010)
22. Weiss, C., Karras, P., Bernstein, A.: Hexastore: sextuple indexing for semantic web data management. PVLDB 1(1), 1008–1019 (2008)

Large-Scale Learning of Relation-Extraction Rules with Distant Supervision from the Web

Sebastian Krause, Hong Li, Hans Uszkoreit, and Feiyu Xu

Language Technology Lab, DFKI
Alt-Moabit 91c, Berlin, Germany
{sebastian.krause,lihong,uszkoreit,feiyu}@dfki.de

Abstract. We present a large-scale relation extraction (RE) system which learns grammar-based RE rules from the Web by utilizing large numbers of relation instances as seed. Our goal is to obtain rule sets large enough to cover the actual range of linguistic variation, thus tackling the long-tail problem of real-world applications. A variant of distant supervision learns several relations in parallel, enabling a new method of rule filtering. The system detects both binary and n-ary relations. We target 39 relations from Freebase, for which 3M sentences extracted from 20M web pages serve as the basis for learning an average of 40K distinctive rules per relation. Employing an efficient dependency parser, the average run time for each relation is only 19 hours. We compare these rules with ones learned from local corpora of different sizes and demonstrate that the Web is indeed needed for a good coverage of linguistic variation.

Keywords: information extraction, IE, relation extraction, RE, rule based RE, web scale IE, distant supervision, Freebase.

1 Introduction

Tim Berners-Lee defines the *Semantic Web* as “a web of data that can be processed directly and indirectly by machines” [4]. Today, there is still a long way to go to reach the goal of a true Semantic Web because most information available on the Web is still encoded in unstructured textual forms, e. g., news articles, encyclopedia like *Wikipedia*, online forums or scientific publications. The research area of *information extraction* (IE) aims to extract structured information from these kinds of unstructured textual data. The extracted information can be instances of concepts such as persons, locations or organizations, or relations among these concepts. *Relation extraction* (RE) deals with the automatic detection of relationships between concepts mentioned in free texts. It can be applied for automatically filling and extending knowledge databases and for semantic annotation of free texts. In recent research, *distant supervision* has become an important technique for data-driven RE (e. g. [15, 16, 22, 32]) because of the availability of large knowledge bases such as *Yago* [21] and *Freebase*[1]. Distant supervision utilizes a large number of known facts of a target domain for automatically labeling mentions of these facts in an unannotated text corpus, hence generating training data.

[1] http://www.freebase.com/

We develop a large-scale RE system that employs Freebase facts as seed knowledge for automatically learning RE rules from the Web in the spirit of distant supervision. The obtained rules can then be applied for the extraction of new instances from new texts. Freebase is a fact database containing some 360 million assertions about 22 million entities such as people, locations, organizations, films, books, etc. We extend the distant supervision approach to RE by combining it with existing means for accommodating relations with arity > 2. To the best of our knowledge, this is the first approach to RE which can learn large-scale grammar-based RE rules for n-ary relations from the Web in an efficient way. We try to learn from the Web as many such rules as possible. For these rules, we adopt the rule formalism of the DARE framework [31], because it accommodates relations of various complexity and is expressive enough to work with different linguistic formalisms, in particular, results of deeper analysis such as dependency structures. When applied to parsed sentences, the learned rules can detect relation mentions, extract the arguments and associate them with their respective roles. Therefore, the results can be directly used as input for a knowledge database. In comparison to statistical-classifier approaches like [15, 16], our approach does not only come up with a web-scale RE system but also delivers the extraction rules as an important knowledge source, which can be reused for question answering, textual entailment and other applications.

Our method is applied to 39 relations from the domains *Awards*, *Business* and *People* modeled in Freebase. About 2.8M instances of these relations were retrieved from Freebase as seed knowledge, from which about 200,000 were turned into Bing queries, resulting in almost 20M downloaded web pages. 3M sentences matched by seed facts were utilized to learn more than 1.5M RE rule candidates. Run time for learning was reduced by parallelization with three server machines (16 cores with 2.4 GHz each; 64 GB RAM). We utilize a very efficient dependency parser called MDParser [24]. In our experiments, it takes around 120 ms to parse one sentence of the average length of 25 words. For each relation, the average run time for the entire rule learning process takes only 19 hours.

Our experiments show that the large number of learned rules make useful candidates of RE rules. These rules produce a higher recall than semi-supervised bootstrapping on a domain-relevant small corpus or distant supervision on a large local corpus. However, precision is hampered by a large number of invalid candidate rules. But many of the invalid rule candidates are learned for multiple relations, even for incompatible ones. Therefore, we use the rule overlap between relations for effective filtering. This technique is a new variant of previously proposed methods, i. e., *counter training* [7, 33] and *coupled learning* [6]. It is better suited for distant supervision learning, since it works directly on the rule sets without needing a confidence feedback of extracted instances.

2 Related Work

Real-world applications often benefit from the extraction of n-ary relations, in particular, in the case of event extraction. Very often more than two arguments of an event are mentioned in a single sentence, e. g., in the following example.

Example 1. Prince Albert has married the former Olympic swimmer Charlene Wittstock in a Catholic ceremony in Monaco.

Here, three arguments of a wedding event are mentioned: the two persons (*Prince Albert*, *Charlene Wittstock*) and the location (*Monaco*). In general, the *binary relation only* approaches (e. g, [17, 19, 20]) do not employ the existing syntactic and semantic structures among $n > 2$ arguments and rely on a later component to merge binary relations into relations of higher complexity (e. g., [14]). As described above and explained in Section 4.2, DARE [31] provides a rule extraction strategy, which allows rules to have more than 2 arguments, when they co-occur in one sentence.

Approaches with surface-oriented rule representation (e. g., [11–13, 19]) prefer to employ shallow linguistic analyses thus circumventing less efficient full syntactic parsing for large-scale RE tasks. These formalisms are robust and efficient but only handle binary relations. They work best for relations whose arguments usually co-occur in close proximity within a sentence and whose mentions exhibit limited linguistic variation. In contrast, systems learning RE rules from syntactic structures such as dependency graphs are able to detect relation arguments spread widely across a sentence (e. g., [31, 34]). However, these approaches are usually applied only to relatively small corpora.

The minimally-supervised *bootstrapping* paradigm takes a limited number of initial examples (relation instances or patterns) and labels free texts during several iterations (e. g., [2, 20, 34]). These approaches often suffer from semantic drift and the propagation of errors across iterations [15]. Furthermore, their performance is strongly dependent on the properties of the data, i. e., on specific linguistic variation in conjunction with redundant mention of facts [23]. In contrast, distant supervision approaches [10, 16, 27, 28, 32] rely on a large amount of trustworthy facts and their performance does not hinge on corpus data properties such as redundancy, since multiple occurrences of the same instance in different sentences are not required.

Closely related to our distant supervision approach is the work described by [15], who train a *linear-regression* classifier on examples derived from mentions of *Freebase* relation instances in a large Wikipedia corpus. They focus on the 109 most populated relations of *Freebase*. The trained classifier works on shallow features such as word sequences and POS tags and on dependency relations between words. To our knowledge, neither [15], nor other existing distant supervision approaches can handle n-ary relations.

Parallel to the above approaches, a new paradigm has emerged under the name of *open IE*. A pioneering example is the *TextRunner* system [3, 35]. In contrast to *traditional* RE systems, they do not target fixed relations, thus being very useful for applications continuously faced with new relation or event types, e. g., online social media monitoring. However, the results of these systems cannot be directly taken for filling knowledge databases, because the semantics of the new relations including the roles of the entities remains unknown.

All ML systems for RE are faced with the problem of estimating the confidence of the automatically acquired information. Some approaches utilize the confidence value of the extracted instances or the seed examples as feedback for evaluation of the rules (e. g., [1, 5, 34]). Many others employ negative examples for detecting wrong rules [7, 33], so-called *counter training*. In [30], negative examples are implicitly generated by utilizing a given set of positive relation instances, which form a *closed world*. [6] introduces *coupled learning*, which learns a coupled collection of classifiers for various relations by taking their logical relationships as constraints for estimating the correctness of newly extracted facts. Our current rule filtering method works directly on rules without making use of any confidence information associated with extracted instances.

3 Target Relations and Essential Type

We decide to conduct our experiments in three domains: *Award*, *Business* and *People*. All three domains contain n-ary relations with $n = 2$ and $n > 2$.

Let t be a named-entity type and let $\mathcal{NE}_t$ be the set containing *all* named entities of type t. Let T be a bag of named-entity types and let $n = |T|$. Then any of our n-ary target relations is a set $\mathcal{R}$ for some T with

$$\mathcal{R} \subseteq \prod\nolimits_{t \in T} \mathcal{NE}_t \,. \tag{1}$$

Table 1. Some of the target relations of the *Award*, *Business* and *People* domains

	Argument Names & Entity Types				
Relation	**Slot 1**	**Slot 2**	**Slot 3**	**Slot 4**	**Slot 5**
award nomination	AWARD award concept®	NOMINEE organization® person	DATE date	WORK creative work	
award honor	AWARD award concept®	WINNER organization® person	DATE date	WORK creative work	
hall of fame induction	HALL OF FAME award concept®	INDUCTEE organization® person	DATE date	-	
organization relationship	PARENT organization®	CHILD organization®	FROM date	TO date	-
acquisition	BUYER organization®	ACQUIRED organization®	DATE date	-	-
company name change	NEW organization®	OLD organization®	FROM date	TO date	-
spin off	PARENT organization®	CHILD organization®	DATE date	-	-
marriage	PERSON A person®	PERSON B person®	CEREMONY location	FROM date	TO date
sibling relationship	PERSON A person®	PERSON B person®	-	-	-
romantic relationship	PERSON A person®	PERSON B person®	FROM date	TO date	-
person parent	PERSON person®	PARENT A person®	PARENT B person	-	-

Derived from the modeling in Freebase, the *marriage* relation can formally be described by:

$$\mathcal{R}_{\text{marriage}} \subseteq \mathcal{NE}_{\mathsf{person}} \times \mathcal{NE}_{\mathsf{person}} \times \mathcal{NE}_{\mathsf{location}} \times \mathcal{NE}_{\mathsf{date}} \times \mathcal{NE}_{\mathsf{date}} \,. \tag{2}$$

Often the first k $(k \geq 2)$ arguments of an relation are *essential arguments*, since conceptually the relation holds between these entities.[2] Then we require these arguments in every text mention of an instance. For example, we require both persons in a *marriage* relation to be mentioned, whereas date and location of the wedding are considered optional, as well as a supplementary divorce date. All relations which share the NE types of their essential arguments are of the same *essential type*.

Table 1 shows some of the targeted relations from the three domains *Award*, *Business* and *People*. Due to space restrictions, we only present a subset of the 39 used relations here. Required (essential) arguments are marked by ®. Relations of the same essential type are grouped by solid horizontal lines. For example, all three relations from the *Award* domain (i. e., *award nomination*, *award honor* and *hall of fame induction*) belong to the same essential type since their first two arguments are of the same NE types: award concept and person/organization. All relation definitions used in this paper were taken from Freebase.

4 Architecture

Figure 1 displays the general workflow of our system. First, a local database of relation instances (so-called *seeds*) is generated. The seeds are used as queries for a web search engine, which returns hits potentially containing mentions of the seeds. The web pages are downloaded and transformed into plain texts. After NER, sentences containing at least the essential seed arguments are collected, which are then processed by the dependency parser. We regard a sentence containing at least the essential arguments as a

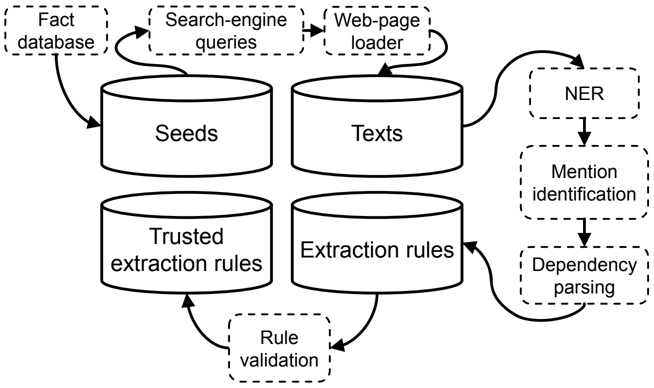


Fig. 1. Data flow of implemented system

[2] Assuming that relations are defined with their most important arguments preceding the others as they actually are in Freebase and most ontologies.

potential mention of a target relation. The parses serve as input for rule learning, which works only on individual sentences. The rule-validation component utilizes information from parallel learning of multiple relations of the same essential types to filter out low-quality rules.

An important design choice is the utilization of the dependency-relation formalism for our rule model. We assume that any given mention of a target-relation instance can be identified by a somehow characteristic pattern in the sentence's underlying dependency graph. This approach has limitations, e. g., it does not cover mentions requiring some kind of semantic understanding (see Section 7), or simply mentions with arguments spread across several sentences. Nevertheless, this methodology is intuitively expressive enough for many mentions. To our knowledge, there exists no systematic investigation of how quantitatively limiting a dependency-formalism based, sentence-restricted approach to RE is.

4.1 Linguistic Annotation

NER in our system is performed by a combination of two components: (a) the *Stanford Named Entity Recognizer*[3] [8] for detection of persons, organizations and locations (extended with our own date recognition), and (b) a simple string fuzzy match via a gazetteer created from the name variations of the seeds' entities as provided by Freebase. In the current system, neither complex entity linking nor coreference resolution are applied in the training phase.

After identification of sentences containing seed mentions, each sentence is processed by the dependency-relation parser *MDParser* (*Multilingual Dependency Parser*)[4] [24]. We choose this parser because it is very fast, while maintaining competitive parsing quality when used in an application, as shown by [25] for the textual entailment task. The parsing results also contain information about part-of-speech tags and word lemmas.

4.2 Rule Learning

We re-use the rule-learning component of the existing DARE system [29, 31]. DARE is a minimally-supervised machine-learning system for RE on free texts, consisting of 1) rule learning (RL) and 2) relation extraction (RE). Starting from a semantic seed (a set of relation instances), RL and RE feed each other in a bootstrapping framework. In our system, we use the RL component for the training phase (Section 5) and the RE part in the evaluation (Section 6). DARE is able to directly handle n-ary relations through its extraction-rule formalism, which models the links between relation arguments using dependency relations.

Consider for example the *marriage* relation from Table 1, which has the arguments PERSONA, PERSONB, CEREMONY, FROM, and TO. Given the seed tuple ⟨*Brad Pitt, Jennifer Aniston, Malibu, 2000, 2005*⟩, the following sentence can be used for rule learning:

[3] *Stanford CoreNLP* (version 1.1.0) from
`http://nlp.stanford.edu/software/corenlp.shtml`

[4] See `http://mdparser.sb.dfki.de/`

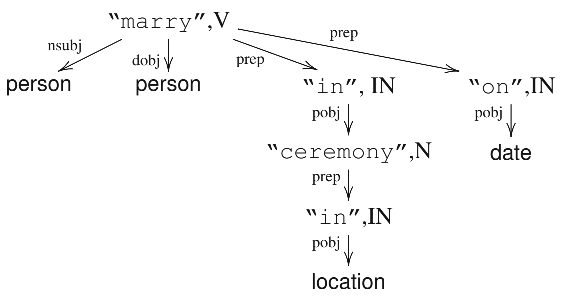


Fig. 2. Dependency parse of Example 2

Rule name :: PersonA_PersonB_Ceremony_From
Rule body ::

$$\begin{bmatrix} \text{head} & (\texttt{"marry"}, \text{V}) \\ \text{dobj} & \begin{bmatrix}\text{head} & \boxed{0}\ \textsf{person}\end{bmatrix} \\ \text{prep} & \begin{bmatrix} \text{head} & (\texttt{"in"}, \text{IN}) \\ \text{pobj} & \begin{bmatrix} \text{head} & (\texttt{"ceremony"}, \text{N}) \\ \text{prep} & \begin{bmatrix} \text{head} & (\texttt{"in"}, \text{IN}) \\ \text{pobj} & \begin{bmatrix}\text{head} & \boxed{1}\ \textsf{location}\end{bmatrix} \end{bmatrix} \end{bmatrix} \end{bmatrix} \\ \text{prep} & \begin{bmatrix} \text{head} & (\texttt{"on"}, \text{IN}) \\ \text{pobj} & \begin{bmatrix}\text{head} & \boxed{2}\ \textsf{date}\end{bmatrix} \end{bmatrix} \\ \text{nsubj} & \begin{bmatrix}\text{head} & \boxed{3}\ \textsf{person}\end{bmatrix} \end{bmatrix}$$

Output :: $\left\langle \boxed{0}\ \text{PERSON A}, \boxed{3}\ \text{PERSON B}, \boxed{1}\ \text{CEREMONY}, \boxed{2}\ \text{FROM}, \text{—} \right\rangle$

Fig. 3. Example rule for the *marriage* relation

Example 2. Brad Pitt married Jennifer Aniston in a private wedding ceremony in Malibu on July 29, 2000.

This sentence is processed by the dependency parser, which outputs a structure like in Figure 2, where the surface strings of the named entities have already been replaced by their respective types in this tree via the NER.

From this dependency tree, DARE learns the rule in Figure 3, which contains four arguments: two married persons plus the wedding location and the starting date of the marriage. DARE additionally learns projections of this rule, namely, rules containing a subset of the arguments, e. g., only connecting the person arguments. This way, a single sentence might lead to the learning of several rules.

5 Web-Scale Rule Learning

Our rule learning consists of two phases: candidate-rule learning and rule filtering. As assumed in distant supervision, when there is a sentence containing the arguments of a relation instance, this sentence is a potential mention of the target relation. Therefore, rules learned from such sentences are also potential rules of the target relation. Because this assumption is not true for all sentences with relation arguments, the resulting rules may be wrong. Hence, they are only *candidate* rules and need further filtering.

5.1 Learning Candidate Rules

Table 2. Number of seeds from Freebase and search hits; statistics about downloaded web pages and documents and sentences containing seed mentions; statistics for rule learning

Relation	**# Seeds**	**# Seeds used**	**# Search hits**	**# Documents w/ a mention**	**# Sentences w/ a mention**	**# Rules**
award nomination	86,087	12,969	1,000,141	14,245	15,499	7,800
award honor	48,917	11,013	1,000,021	50,680	56,198	40,578
hall of fame induction	2,208	2,208	443,416	29,687	34,718	17,450
organization relationship	219,583	70,946	1,000,009	37,475	51,498	28,903
acquisition	1,768	1,768	308,650	40,541	71,124	50,544
company name change	1,051	1,051	124,612	8,690	10,516	6,910
spin off	222	222	32,613	3,608	5,840	4,798
marriage	16,616	6,294	1,000,174	211,186	381,043	176,949
sibling relationship	8,246	8,246	914,582	130,448	186,228	69,596
romantic relationship	544	544	280,508	82,100	172,640	74,895
person parent	23,879	3,447	1,000,023	148,598	213,869	119,238
avg. of 39 relations	72,576	6,095	635,927	60,584	73,938	41,620

In the following, we describe the experimental results of our training phase. Table 2 provides statistics for this phase. For the 39 target relations, 2.8M relation instances were extracted from Freebase (column "# Seeds"). For each relation, we tried to find 1M web documents using the search engine *Bing*[5] (column "# Search hits"), resulting in more than 20M downloaded documents in total for all relations. Note that for some relations, finding 1M web documents required only a subset of the relation instances retrieved from Freebase, while for other relations even utilizing all relation instances was not sufficient for getting 1M web documents. This explains the difference in numbers between columns "# Seeds" and "# Seeds used".

The downloaded web documents were subsequently processed by NER and sentence segmentation. Given sentences with their NE annotations, the sentences with mentions of seeds are identified. The mentions of seeds occur in a relatively small fraction of the downloaded web documents (around 10 %), as shown in column "# Documents w/ a

[5] http://www.bing.com

mention". Reasons for that are 1) seed arguments being spread across sentence borders, 2) NER errors or 3) a wrong (non-English) language of the web document.

The final training corpus contains for each relation on average 74k sentences with mentions of seed instances, i. e., a total of around 3M sentences (column "# Sentences w/ a mention"). All of these mentions include at least the respective relation's essential arguments. On average, around 40k distinct rules were learned per relation (column "# Rules").

The total system runtime per relation was on average around 19 hours, with the processing being distributed on three server machines (16 cores with 2.4 GHz each; 64 GB RAM). The parallelization was accomplished naively by chunking the data according to the respective source seed. Of the pipeline's main processing phases, the search-engine querying and the document download with subsequent text extraction were the most time-consuming ones, with on average 6 hours 17 minutes per relation and 8 hours 40 minutes per relation, respectively. The mention-finding step (including NER) took 3 hours 11 minutes for each relation, the final dependency parsing and rule learning on average only 40 minutes per relation.

5.2 Rule Filtering

Whenever two relations are of the same essential type, they may share some same relation instances, in particular, for the required arguments, for example, the same two persons might be involved in various relations such as marriage and romantic relations. This can be for good reasons, if the relations overlap or if the relevant expressions of the language are ambiguous. Most rules learned for two or more relations, however, are not appropriate for one or both relations. Rules might be learned for wrong relations because of erroneous NER & dependency parsing, false seed facts and false mentions. Especially when a rule is learned for two disjoint relations, something must be wrong. Either the rule exhibits a much higher frequency for one of the two relations, then it can be safely deleted from the other, or the rule is wrong for both relations. Figure 4

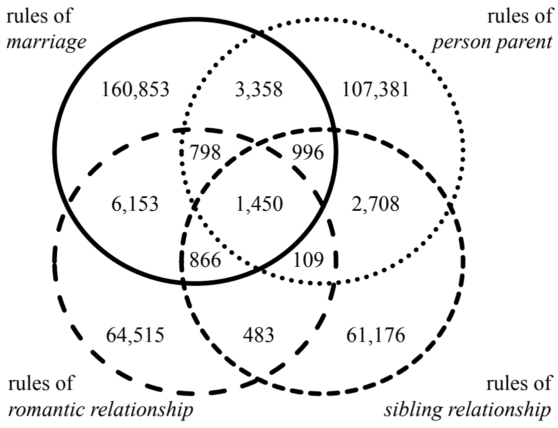


Fig. 4. Euler diagram showing numbers of rules learned for four *People* relations. Missing zones: *person parent / romantic relationship* (408); *marriage / sibling relationship* (1,808).

shows intersections of the sets of learned rules for four relations of the same essential type in the *People* domain: *marriage*, *romantic relationship*, *person parent*, and *sibling relationship*. Rules in the intersections either express one of the displayed relations or a non-displayed relation or no specific semantic relation at all.

We propose a general and parametrizable filtering strategy using information about the applicability of a rule w. r. t. other relations of the same essential type. If a rule occurs significantly more often in a relation $\mathcal{R}$ than in another relation $\mathcal{R}'$, this rule most probably belongs to $\mathcal{R}$. Let $f_{r,\mathcal{R}}$ be the frequency of rule r in relation $\mathcal{R}$ (i. e., the number of sentences for $\mathcal{R}$ from which r has been learned) and let $R_{\mathcal{R}}$ be the set of learned rules for $\mathcal{R}$. Then the relative frequency of r in $\mathcal{R}$ is defined as:

$$rf_{r,\mathcal{R}} = \frac{f_{r,\mathcal{R}}}{\sum\limits_{r' \in R_{\mathcal{R}}} f_{r',\mathcal{R}}} \tag{3}$$

Next, we define the first component of our filter. Let $\mathbb{R}$ be a set of relations of the same essential type. The rule r is *valid* for the relation $\mathcal{R} \in \mathbb{R}$ if the relative frequency of r in $\mathcal{R}$ is higher than its relative frequencies for all other relations in $\mathbb{R}$:

$$valid^{\mathcal{R}}_{inter}(r) = \begin{cases} \textit{true} & \text{if } \forall \mathcal{R}' \in \mathbb{R} \backslash \{\mathcal{R}\} : rf_{r,\mathcal{R}} > rf_{r,\mathcal{R}'} \\ \textit{false} & \text{otherwise} \end{cases} \tag{4}$$

The second component is a heuristic which only filters on the frequency of a rule w. r. t. a single relation:

$$valid^{\mathcal{R}}_{freq}(r) = \begin{cases} \textit{true} & \text{if } f_{r,\mathcal{R}} \geq x, \text{where } x \geq 1 \\ \textit{false} & \text{otherwise} \end{cases} \tag{5}$$

With this filter, we ensure that in addition to the relative frequency, there is also enough evidence that r belongs to $\mathcal{R}$ from an absolute point of view. We merge the two components into our final filter, later referred to as the *combined filter*:

$$valid^{\mathcal{R}}_{c}(r) = valid^{\mathcal{R}}_{freq}(r) \wedge valid^{\mathcal{R}}_{inter}(r) \tag{6}$$

Note that all rules that do not contain any content words such as verbs, nouns or adjectives will be deleted before the actual rule filtering takes place. In addition to the frequency heuristic, we also experimented with other features, such as the arity of rules and the length of rules' source sentences. However, their general performance was not superior to the frequency of a rule.

6 Testing and Evaluation

Since we are in particular interested in the recall and coverage performance of our learned rules, we are more dependent on the gold-standard data than precision-driven evaluations as presented in [15], where they evaluate manually the top 100 or 1000 extracted instances of the most popular relations. The ACE 2005 corpus [26] is too sparse for our evaluation goal, for example, there are only 14 mentions containing the essential person arguments for the *marriage* relation. The annotation of the MUC-6 corpus [9] is document-driven and does not provide direct links between relation arguments and sentences. Therefore, we had to prepare a new gold-standard test corpus annotated with relations and their arguments sentence-wise. Because of high annotation costs, we decided to focus on one relation, namely, the *marriage* relation. On this new gold-standard corpus, we compare our system's web rules against rules learned with the basic DARE system.

In order to know the impact of training corpus size on the coverage of the learned rules in the distant supervision approach, we also compare the recall performance of the rules learned from the Web with rules learned from local corpora of two different sizes. All learned rules are tested against the New York Times part of the English Gigaword 5 corpus [18].

6.1 Learned Marriage Relation Rules

The *marriage* relation has five arguments: PERSONA, PERSONB, CEREMONY, FROM, and TO. A candidate rule must extract at least the two person arguments. The distribution of the rules with respect to their arities is depicted in Table 3. Although many rules are binary, there are more than 20 % of the total rules with arities > 2 (more than 30k). It demonstrates that it is important to learn n-ary rules for the coverage.

Table 3. Distribution of *marriage* rules across arities. "Avg." – Average, "Med." – Median.

Arity	#Rules	Min.Freq.	Avg.Freq.	Med.Freq.	Max.Freq.
2	145,598	1	3.21	1	64,015
3	26,294	1	2.90	1	2,655
4	4,350	1	3.07	1	603
5	40	1	1.40	1	10

6.2 Evaluation on Gold-Standard Corpus

Our gold-standard corpus, dubbed *Celebrity-Gold*, contains crawled news articles from the *People* magazine.[6] This corpus consists of 25,806 sentences with 259 annotated mentions of *marriage*. Out of curiosity, we compare the web-based learning to the bootstrapping approach using the same system components and the same seed (6,294 relation instances). The learning corpus for bootstrapping, dubbed *Celebrity-Training*, is of the same kind and size as Celebrity-Gold. Compared to the 176,949 candidate rules from the Web, the bootstrapping system learned only 3,013 candidate rules.

The learned rules are then applied to Celebrity-Gold for evaluation. It tuns out that our web-based system achieves much higher recall than the bootstrapping system: 49.42 % vs. 30.5 %. As we know, the learned web rules are in fact only candidates for RE rules. Therefore, the baseline precision is relatively low, namely, 3.05 %. Further processing is needed to filter out the wrong rules. Nevertheless, investigating the recall at this stage

[6] http://www.people.com/

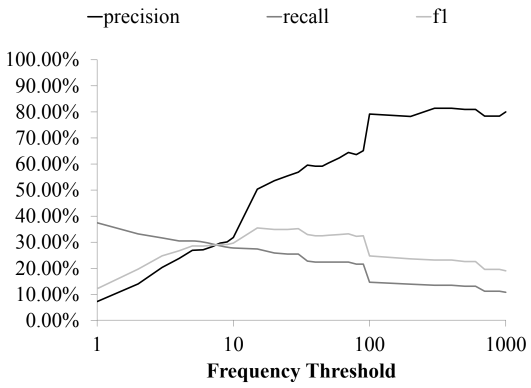


Fig. 5. Performance of web rules after filtering. X-axis: frequency thresholds

is very important because even an excellent rule filtering might produce below-average results if there were not enough correct rules to separate from wrong ones during the filtering phase.

Figure 5 depicts the extraction performance after the combined filter $valid_c^{\mathcal{R}}(r)$ is applied to the learned *marriage* rules. The precision improves considerably, in particular, grows with high frequency. The best f-measure can be obtained by setting the frequency to 15 with a precision of around 50% and a recall of around 28%.

6.3 Evaluation with Different Corpus Sizes

After the encouraging results on the small-sized Celebrity-Gold corpus, we evaluated our rules by applying them to a larger corpus, the NYT subset of the English Gigaword 5 corpus (abbr. NYT). Because there is no gold-standard annotation of the *marriage* relation available for this corpus, we use two alternative validation methods: (a) manual checking of all mentions detected by our rules in a random partition of NYT (100,000 sentences) and (b) automatic matching of extracted instances against the Freebase facts about *marriage*s. Note that before RE was performed, we removed all web training sentences from NYT, to avoid an overlap of training and test data.

The performance of the web rules is compared to rules learned on two local corpora in a distant-supervision fashion. The first corpus is the Los Angeles Times/Washington Post part of the Gigaword corpus (abbr. LTW). The second local corpus for rule learning

Table 4. Statistics about corpus sizes and rule learning

Corpus	**# Docs**	**# Sentences**	**# Seeds w/ match**	**# Generated training sentences**	**# Rules learned**
Web (train.)	873,468	81,507,603	5,993	342,895	176,949
LTW (train.)	411,032	13,812,110	1,382	2,826	1,508
Celebrity-Training (train.)	150	17,100	76	204	302
NYT (test)	1,962,178	77,589,138	–	–	–

is the corpus used for bootstrapping in Section 6.2: Celebrity-Training. Here, only the rules learned in the first bootstrapping iteration were employed for relation extraction to allow for better comparison. For both local training corpora, the same seed set as for the web learning was used (i. e., 6,294 instances). Table 4 shows statistics about the corpora and provides information about the learned rules.

Table 5 shows the extraction results of the different rule sets on NYT. The web candidate rules without rule filtering find the highest number of positive marriage mentions of Freebase instances in the corpus, namely, 1,003. This experiment confirms the hypothesis that the extraction coverage of the learned rules increases with the size of the training corpus. After the rule filtering, the web system has improved the precision effectively without hurting recall too much. Note that different kinds of rule filtering may be applied also to the rules learned from Celebrity-Training and LTW. Because the focus of this research is web learning, we only show the results for the web system here.

Table 5. Extraction results on NYT corpus for rules from distant-supervision learning on different corpus sizes. "# Freebase" is short for "# Extracted instances confirmed as correct by Freebase".

			Mentions in sample		
Source of rules	**Filter applied**	**# Freebase**	**# correct**	**# wrong**	**Precision**
Web	–	1,003	76	1,747	4.17 %
LTW	–	721	47	414	10.20 %
Celebrity-Training	–	186	7	65	9.72 %
Web	$valid^{\mathcal{R}}_{inter}(r)$	884	69	869	7,36 %
Web	$valid^{\mathcal{R}}_{c}(r)$, with $x = 15$	627	52	65	44.44 %
Web	$valid^{\mathcal{R}}_{c}(r)$, with $x = 30$	599	51	18	73.91 %

7 Error Analysis

Section 6.2 states that the learned rules covered 49.42 % of the gold-standard mentions in Celebrity-Gold. In this section, we analyze why the system missed the other half of mentions. Table 6 shows the results of a manual investigation of the false negatives of our system on Celebrity-Gold.

Because our system operates on top of NER and parsing results, it heavily depends on correct output of these preprocessing tools. On 41.22 % of false negatives, flawed NER rendered annotated mentions undetectable for extraction rules, even if we had learned *all* possible rules in the training phase. Example errors include unrecognized person entities and broken coreference resolution. Even worse, the parser returned for 59.54 % of the false negatives dependency graphs with errors on the paths between mention arguments, again stopping extraction rules from finding the mentions.

To approximate the system's recall in a setting with perfect linguistic preprocessing, we removed the mistakenly annotated mentions and fixed the errors in NER and parsing. We then reassessed whether a matching extraction rule had been learned in the training phase. Surprisingly, for about half of the remaining false negatives an extraction rule had actually been learned, i. e., the system's main problem is the unreliability of linguistic

Table 6. Analysis of false negatives (abbr.: "fn.") on Celebrity-Gold

	% of fn.	**# of fn.**
Total	100.00	131
Annotation error	4.58	6
Linguistic preprocessing error[7]	84.73	111
• NER error	41.22	54
• Parsing error	59.54	78
Total	100.00	125
Matching rule actually learned	50.40	63
No matching rule learned	27.20	34
Semantic understanding required	22.40	28

[7]Note: A fn. might suffer from errors in both NER and parsing.

preprocessing, not a lack of coverage in its rules. In other words, the recall value stated in Section 6.2 would have been about 25 percentage points higher, if NER and parsing had worked perfectly.

An error class that cannot be attributed to accuracy deficits of linguistic processing contains sentences that require semantic understanding. These sentences mention an instance of the *marriage* relation, but in an ambiguous way or in a form were the relation is understood by a human, although it is not directly represented in the sentence's structure. The following sentence from the gold-standard corpus is a typical example for this class since the syntactic dependencies do not link "husband" directly to "Ruiz."

Example 3. "... that sounded good to a tired mom like me," says Ruiz, 34, who has two children, James, 8, and Justine, 6, with husband Donald, 42, ...

For a human reader, it is obvious that the phrase "*with husband Donald*" belongs to the person *Ruiz*, because of her mentioning of mother role in the family context. However, attaching the phrase to *Justine* might very well be a reasonable decision for a parser. This becomes clearer when the sentence is slightly changed:

Example 4. "... that sounded good to a tired mom like me," says Ruiz, 34, who awaits her guests, James, 33, and Justine, 35, with husband Donald, 42.

Here, even a human reader cannot decide whose husband *Donald* is. Another example is the following sentence:

Example 5. Like countless mothers of brides, Ellen Mariani smiled until her cheeks ached as she posed for wedding pictures with her daughter Gina, 25, and newly minted son-in-law Christopher Bronley, 22, on Saturday, Sept. 15.

Here it is not clear from the structure that *Christopher Bronley* and *Gina* are spouses. Inference is needed to entail that *Gina* is married to *Christopher Bronley* because she is the daughter of *Ellen Mariani*, who in turn is the mother-in-law of *Christopher Bronley*.

8 Conclusion and Future Work

Our system for the extraction of n-ary relations exploits the Web for training. After achieving an improvement of recall, precision was raised by a rule-filtering scheme that exploits negative evidence obtained from the applicability of a rule to other relations of the same essential type. The parallel learning of several relations hence proved to be beneficial. We demonstrate that web-scale distant-supervision based rule learning can achieve better recall and coverage than working with local large corpora or bootstrapping on small local corpora. Furthermore, rules with arities > 2 are useful resources for RE.

The error analysis clearly indicates that recall could be much higher if named entity recognition (NER) and parsing worked more accurately. As a consequence of this insight, we will concentrate on the improvement of NER using the rapidly growing resources on the Web and on the adaptation of parsers to the needs of RE, by experimenting with specialized training and parse re-ranking. Another direction of future research will be dedicated to the incorporation of more sophisticated methods for rule filtering. A first step is to exploit additional information on the relationships among the target relations for estimating the validity of rules, another strategy is to re-estimate the confidence of rules during the application phase utilizing constraints derived from the domain model.

Acknowledgments. This research was partially supported by the German Federal Ministry of Education and Research (BMBF) through the project Deependance (contract 01IW11003), by the German Research Foundation (DFG) through the Cluster of Excellence on Multimodal Computing and Interaction (M2CI), and by Google Inc through a Faculty Research Award granted in July 2012.

References

1. Agichtein, E.: Confidence estimation methods for partially supervised information extraction. In: Ghosh, J., Lambert, D., Skillicorn, D.B., Srivastava, J. (eds.) SDM 2006. SIAM (2006)
2. Agichtein, E., Gravano, L.: Snowball: extracting relations from large plain-text collections. In: Fifth ACM Conference on Digital Libraries, pp. 85–94. ACM (2000)
3. Banko, M., Cafarella, M.J., Soderland, S., Broadhead, M., Etzioni, O.: Open information extraction from the Web. In: Veloso, M.M. (ed.) IJCAI 2007, pp. 2670–2676 (2007)
4. Berners-Lee, T.: Weaving the Web: The Original Design and Ultimate Destiny of the World Wide Web by Its Inventor. HarperCollins, New York (1999)
5. Brin, S.: Extracting Patterns and Relations from the World Wide Web. In: Atzeni, P., Mendelzon, A.O., Mecca, G. (eds.) WebDB 1998. LNCS, vol. 1590, pp. 172–183. Springer, Heidelberg (1999)
6. Carlson, A., Betteridge, J., Hruschka Jr., E.R., Mitchell, T.M.: Coupling semi-supervised learning of categories and relations. In: NAACL HLT 2009 Workshop on Semi-supervised Learning for Natural Language Processing, pp. 1–9 (2009)
7. Etzioni, O., Cafarella, M., Downey, D., Popescu, A.M., Shaked, T., Soderland, S., Weld, D.S., Yates, A.: Unsupervised named-entity extraction from the Web: An experimental study. Artif. Intell. 165, 91–134 (2005)
8. Finkel, J.R., Grenager, T., Manning, C.: Incorporating non-local information into information extraction systems by Gibbs sampling. In: ACL 2005 (2005)
9. Grishman, R., Sundheim, B.: Message understanding conference - 6: A brief history. In: COLING 1996, pp. 466–471 (1996)
10. Hoffmann, R., Zhang, C., Weld, D.S.: Learning 5000 relational extractors. In: ACL 2010, pp. 286–295 (2010)

11. Hovy, E.H., Kozareva, Z., Riloff, E.: Toward completeness in concept extraction and classification. In: EMNLP 2009, pp. 948–957 (2009)
12. Kozareva, Z., Hovy, E.H.: A semi-supervised method to learn and construct taxonomies using the Web. In: EMNLP 2010, pp. 1110–1118 (2010)
13. Kozareva, Z., Riloff, E., Hovy, E.H.: Semantic class learning from the Web with hyponym pattern linkage graphs. In: ACL 2008, pp. 1048–1056 (2008)
14. McDonald, R., Pereira, F., Kulick, S., Winters, S., Jin, Y., White, P.: Simple algorithms for complex relation extraction with applications to biomedical IE. In: ACL 2005 (2005)
15. Mintz, M., Bills, S., Snow, R., Jurafsky, D.: Distant supervision for relation extraction without labeled data. In: Su, K.Y., Su, J., Wiebe, J. (eds.) ACL/IJCNLP 2009, pp. 1003–1011 (2009)
16. Nguyen, T.V.T., Moschitti, A.: End-to-end relation extraction using distant supervision from external semantic repositories. In: ACL 2011, Short Papers, pp. 277–282 (2011)
17. Pantel, P., Ravichandran, D., Hovy, E.: Towards terascale semantic acquisition. In: COLING 2004 (2004)
18. Parker, R., et al.: English Gigaword Fifth Edition. Linguistic Data Consortium, Philadelphia (2011)
19. Pasca, M., Lin, D., Bigham, J., Lifchits, A., Jain, A.: Names and similarities on the web: Fact extraction in the fast lane. In: ACL/COLING 2006 (2006)
20. Ravichandran, D., Hovy, E.H.: Learning surface text patterns for a question answering system. In: ACL 2002, pp. 41–47 (2002)
21. Suchanek, F.M., Kasneci, G., Weikum, G.: YAGO: A large ontology from Wikipedia and WordNet. J. Web. Semant. 6, 203–217 (2008)
22. 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: Proceedings of the Fourth Text Analysis Conference (2011)
23. 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)
24. Volokh, A.: MDParser. Tech. rep., DFKI GmbH (2010)
25. Volokh, A., Neumann, G.: Comparing the benefit of different dependency parsers for textual entailment using syntactic constraints only. In: SemEval-2 Evaluation Exercises on Semantic Evaluation PETE (2010)
26. Walker, C., Strassel, S., Medero, J., Maeda, K.: ACE 2005 multilingual training corpus. Linguistic Data Consortium, Philadelphia (2006)
27. Weld, D.S., Hoffmann, R., Wu, F.: Using Wikipedia to bootstrap open information extraction. SIGMOD Record 37, 62–68 (2008)
28. Wu, F., Hoffmann, R., Weld, D.S.: Information extraction from Wikipedia: moving down the long tail. In: KDD 2009, pp. 731–739 (2008)
29. Xu, F.: Bootstrapping Relation Extraction from Semantic Seeds. Ph.D. thesis, Saarland University (2007)
30. Xu, F., Uszkoreit, H., Krause, S., Li, H.: Boosting relation extraction with limited closed-world knowledge. In: COLING 2010, Posters, pp. 1354–1362 (2010)
31. Xu, F., Uszkoreit, H., Li, H.: A seed-driven bottom-up machine learning framework for extracting relations of various complexity. In: ACL 2007 (2007)
32. Xu, W., Grishman, R., Zhao, L.: Passage retrieval for information extraction using distant supervision. In: IJCNLP 2011, pp. 1046–1054 (2011)
33. Yangarber, R.: Counter-training in discovery of semantic patterns. In: ACL 2003. pp. 343–350 (2003)
34. Yangarber, R., Grishman, R., Tapanainen, P.: Automatic acquisition of domain knowledge for information extraction. In: COLING 2000, pp. 940–946 (2000)
35. Yates, A., Cafarella, M., Banko, M., Etzioni, O., Broadhead, M., Soderland, S.: TextRunner: open information extraction on the Web. In: HLT-NAACL 2007, Demonstrations, pp. 25–26 (2007)

The Not-So-Easy Task of Computing Class Subsumptions in OWL RL

Markus Krötzsch

Department of Computer Science, University of Oxford, UK
markus.kroetzsch@cs.ox.ac.uk

Abstract. The lightweight ontology language OWL RL is used for reasoning with large amounts of data. To this end, the W3C standard provides a simple system of deduction rules, which operate directly on the RDF syntax of OWL. Several similar systems have been studied. However, these approaches are usually complete for instance retrieval only. This paper asks if and how such methods could also be used for computing entailed subclass relationships. Checking entailment for arbitrary OWL RL class subsumptions is co-NP-hard, but tractable rule-based reasoning is possible when restricting to subsumptions between atomic classes. Surprisingly, however, this cannot be achieved in any RDF-based rule system, i.e., the W3C calculus cannot be extended to compute all atomic class subsumptions. We identify syntactic restrictions to mitigate this problem, and propose a rule system that is sound and complete for many OWL RL ontologies.

1 Introduction

The lightweight ontology language OWL RL [16] is widely used for reasoning with large amounts of data, and many systems support query answering over OWL RL ontologies. Commercial implementations of (parts of) OWL RL include Oracle 11g [12], OWLIM [2], Virtuoso [4], and AllegroGraph [5].

What makes OWL RL so appealing to users and implementers alike are its favourable computational properties. As for all three lightweight profiles of OWL 2, typical reasoning tasks for OWL RL can be solved in polynomial time. Possibly even more important, however, is the fact that this low complexity can be achieved by relatively straightforward algorithms that perform bottom-up, rule-base materialisation of logical consequences until saturation. While it is also possible to use similar algorithms for OWL EL [14], a simple rule-based algorithm for OWL RL is already given in the W3C specification [16]. Various similar rule sets have been published for fragments of OWL RL (e.g., [9,7,20]).

Another advantage of these rule systems is that they operate directly on the RDF serialisation of OWL. OWL RL reasoning thus can easily be implemented on top of existing RDF databases that support some form of production rules to infer additional information. Basic rule-matching capabilities are found in most RDF stores, since they are similar to query matching. Indeed, *SPARQL rules* have been proposed as a natural rule extension of the SPARQL query language that can fully express the OWL RL rule system [19]. It is important that neither value invention (*blank node creation*) nor non-monotonic negation are needed in OWL RL, as both features complicate rule evaluation significantly [19].

A common strategy for evaluating larger amounts of OWL RL data is to separate terminological information (schema-level axioms) from assertional information (facts), since the latter are typically significantly larger [13,8,20]. To further reduce the rules that need to be applied to the data, it would be useful to pre-compute terminological inferences, especially all subclass relationships.

Unfortunately, it is not known how to do this. To the best of our knowledge, no polynomial time algorithm has been published for computing schema entailments in OWL RL. As shown in Section 2, the W3C rule system is not complete for computing class subsumptions, and, more problematically, it is not possible to compute all class subsumptions without taking assertions into account.

However, it is still possible to obtain RDF-based rule systems that can discover more entailments than the W3C rules. Using various abbreviations introduced in Section 3, we present one such algorithm in Section 4. We identify ObjectHasValue as the main obstacle – if it does not occur in superclasses, our algorithm can compute all class subsumptions. In Section 5, we take a look at RDF-based rules and show how our results transfer to this setting.

In Section 6, we discuss the problem of computing class subsumptions in unrestricted OWL RL ontologies, where ObjectHasValue is allowed in superclasses. It is still possible to use polynomial time rule systems for this case, but it turns out that there is no *RDF-based* rule system for this task. This surprising result shows an inherent limitation of the expressive power of RDF-based rules.

Most of our presentation is based on the Functional-Style Syntax of OWL [17]. This yields a more concise presentation (compared to the RDF serialisation) and still is close to the actual language. We assume basic familiarity with the syntax and semantics of OWL on the level of the *OWL Primer* [6]. If not stated otherwise, we generally consider the Direct Semantics of OWL, but we also mention some results about the RDF-based Semantics. When writing OWL entities in axioms, we tacitly use prefixes for suggesting abbreviated IRIs. Some proofs are omitted for reasons of space; they are found in a technical report [15].

2 Challenges of Schema Reasoning in OWL RL

Before we focus on the task of computing class subsumptions for OWL RL, it is worth pointing out some limitations and challenges that do not seem to be well known, even among practitioners and implementers.

Checking Entailment of OWL RL Axioms is Not Tractable. The W3C specification mentions that *Class Expression Subsumption* in OWL RL is PTime-complete w.r.t. the size of the ontology [16, Section 5]. This might evoke the impression that one could check in polynomial time whether an OWL RL class inclusion axiom is entailed by an OWL RL ontology. This is not the case.[1]

[1] The 2012 update of the OWL 2 specification will correct this; see Section 7.

Proposition 1. *Given an OWL RL ontology $\mathcal{O}$ and an OWL RL* SubClassOf *axiom A, checking whether $\mathcal{O}$ entails A is co-*NP*-hard.*

Proof. We show this by reducing 3SAT unsatisfiability to OWL RL entailment checking. An instance of the 3SAT problem is a set of propositional clauses $\{(L_{11} \vee L_{12} \vee L_{13}), \ldots, (L_{n1} \vee L_{n2} \vee L_{n3})\}$, where each L_{ij} is a propositional variable or a negated propositional variable. The question whether the conjunction of these clauses is satisfiable is NP-complete. For each propositional variable p, we introduce two new class names T_p and F_p. To each literal L_{ij}, we assign a class name $c(L_{ij})$ as follows: if $L_{ij} = p$, then $c(L_{ij}) := T_p$; if $L_{ij} = \neg p$, then $c(L_{ij}) := F_p$. For every clause $(L_{i1} \vee L_{i2} \vee L_{i3})$, we define a class expression C_i as ObjectUnionOf($c(L_{11})$ $c(L_{12})$ $c(L_{13})$). Now let A be the axiom SubClassOf(ObjectIntersectionOf($C_1 \ldots C_n$) owl:Nothing), and let $\mathcal{O}$ be the ontology that consists of the axioms DisjointClasses(T_p F_p) for every propositional variable p. Clearly, A is an OWL RL axiom and $\mathcal{O}$ is an OWL RL ontology. However, $\mathcal{O}$ entails A if and only if the given instance of 3SAT has *no* solution. Indeed, if A is not entailed, then $\mathcal{O}$ has a model where an individual e is an instance of each of the class expression C_i. We can construct a propositional truth assignment as follows: if e is an instance of T_p, then p is mapped to true; otherwise p is mapped to false. It is easy to see that this is a solution to the 3SAT instance, since e cannot be an instance of T_p and F_p for any p. □

Another way to find hardness proofs is to use DataSomeValuesFrom in the subclass, together with datatypes such as XML Schema boolean, which is admissible in OWL RL subclasses. Moreover, similar problems can be found for other axiom types; e.g., checking entailment of hasKey axioms is also intractable.

These problems are hardly surprising from a logical perspective, since checking the entailment of A from $\mathcal{O}$ is equivalent to checking the consistency of $\mathcal{O} \cup \{\neg A\}$, where $\neg A$ is the negation of the axiom A (as a logical formula). For the latter to be in OWL RL, one needs to impose different syntactic restrictions on A. The check is then possible in PTime. A particularly relevant case where this is possible is that A is a subclass relationship between two class names. This is the task that we will focus on in the remainder of this work.

The W3C Rule System is Not Complete for Class Subsumption Checking. The W3C specification states a completeness theorem for its rule system, which asserts completeness only for entailment of assertional axioms. However, the rule system contains a number of rules that would not be necessary to achieve this especially in Table 9, entitled *The Semantics of Schema Vocabulary.* This might lead to the wrong impression that the rules can infer all schema axioms, or at least all class subsumptions. The following example illustrates that this is wrong.

Example 1. We consider an ontology of three axioms:

SubClassOf(:A :B) (1)

SubClassOf(:A :C) (2)

SubClassOf(ObjectIntersectionOf(:B :C) :D) (3)

This ontology clearly entails SubClassOf(:A :D). However, this is not entailed by the W3C rule system. The only entailment rules that refer to ObjectIntersectionOf are rules *cls-int1*, *cls-int2*, and *scm-int* in [16]. The former two rules are only applicable to individual instances. Rule *scm-int* can be used to infer SubClassOf(ObjectIntersectionOf(:B :C) :B) and SubClassOf(ObjectIntersectionOf(:B :C) :C) – the rule can be viewed as a schema-level version of *cls-int2*. However, there is no rule that corresponds to *cls-int1* on the schema level, so one can not infer SubClassOf(:A ObjectIntersectionOf(:B :C)).

This example extends to many other types of axioms. For example, one cannot infer all entailed property domains or ranges if some class subsumptions have been missed.

Assertions Can Not be Ignored when Checking Class Subsumptions. Since many OWL RL ontologies are dominated by assertional axioms (i.e., data), it would be useful if this part of the ontology would not be relevant if one is only interested in computing class subsumptions. This cannot work in general, since assertions can cause the ontology to become inconsistent, which in turn leads to the entailment of arbitrary class subsumptions. However, even for consistent ontologies, assertions cannot be ignored when computing class subsumptions, as shown in the next example.

Example 2. We consider an ontology of three axioms:

InstanceOf(:B :b) (4)

SubClassOf(:A ObjectHasValue(:P :b)) (5)

SubClassOf(ObjectSomeValuesFrom(:P :B) :C) (6)

This ontology entails SubClassOf(:A :D): every instance of :A has a :P successor :b (5), that is an instance of :B (4); thus the subclass in (6) is a superclass of :A. Without (4) this entailment would be missed.

The relevant assertional information in this example was directly given, but it is clear that it could also be the result of more complicated reasoning. Therefore, it is not possible in general to compute all class subsumptions of an ontology without also computing a significant amount of fact entailments as well. Theorem 3 in Section 4 below identifies a case where assertions can be ignored.

3 A Simpler OWL RL

OWL is a very rich language that provides many redundant syntactic constructs for the convenience of ontology engineers. When specifying a rule system for OWL RL, this abundance of syntax precludes a concise presentation – the W3C calculus already counts 78 rules. To avoid this problem, we introduce various simplification rules that allow us to restrict to a much smaller number of features.

Syntactic simplifications can affect the semantics of an ontology. On the one hand, they may introduce auxiliary vocabulary symbols that had not been defined by the original ontology. On the other hand, any syntactic transformation

$$\mathsf{R_{ca1}}\ \frac{\mathsf{ClassAssertion(}\ C\ a\ \mathsf{)}}{\mathsf{SubClassOf(\ ObjectOneOf(}a\mathsf{)}\ C\ \mathsf{)}} \qquad \mathsf{R_{ca2}}\ \frac{\mathsf{SubClassOf(\ ObjectOneOf(}a\mathsf{)}\ C\ \mathsf{)}}{\mathsf{ClassAssertion(}\ C\ a\ \mathsf{)}}$$

$$\mathsf{R_{pc1}}\ \frac{\mathsf{ObjectPropertyAssertion(}\ P\ a\ b\ \mathsf{)}}{\mathsf{SubClassOf(\ ObjectOneOf(}a\mathsf{)\ \ ObjectSomeValuesFrom(}\ P\ \mathsf{ObjectOneOf(}b\mathsf{)\ )\ )}}$$

$$\mathsf{R_{pc2}}\ \frac{\mathsf{SubClassOf(\ ObjectOneOf(}a\mathsf{)\ \ ObjectSomeValuesFrom(}\ P\ \mathsf{ObjectOneOf(}b\mathsf{)\ )\ )}}{\mathsf{ObjectPropertyAssertion(}\ P\ a\ b\ \mathsf{)}}$$

Fig. 1. Rules for expressing assertions as class subsumptions

of expressions has an impact on the RDF-based Semantics of OWL, which entails only axioms about expressions that are syntactically present in the ontology. Adding a new expression, even if tautological, thus changes entailments. However, we expect applications that rely on the RDF-based Semantics to tolerate this effect. Alternatively, it is possible to view our syntactic simplifications as a mere abbreviation scheme for a much larger number of rules.

Lists in Axioms. We generally assume that all lists of classes or properties in OWL axioms have been binarised, that is, broken down into lists of length two. This is always possible by introducing additional axioms; we omit the details of this frequently used technique. We point out that this simplification is least essential for our rules. It is easy to generalise all rules we state for binary lists to lists of arbitrary length.

Datatypes and Data Properties. We omit all features related to datatypes from our presentation. It is not hard to add them. In essence, datatypes in OWL RL behave like classes for which certain subsumptions are given upfront (e.g., *decimal* is a superclass of *integer*), and for which some disjointness axioms are known (e.g., *rational* is disjoint from *string*). Likewise, datatype literals behave like individual names for which the membership in some class (i.e., datatype) is known. This auxiliary information can be added when loading an ontology. Thereafter, datatypes can be treated like classes, using the same rule system.

Assertions as Class Inclusions. Class and property assertions can be expressed as class inclusion axioms, and indeed many rules for assertions are just special forms of the rules needed for terminological axioms. To avoid unnecessary repetition, we generally use the respective forms interchangeably. This is formalised by the rules in Fig. 1. The rules are applied top to bottom: whenever the axioms above the line are found, the axioms below the line are added. C denotes a class expression (not necessarily a class name), and P denotes an object property expression. We use a and b for individual names (IRIs).

We will make use of the shorter syntactic form of assertions whenever possible. Note that the class subsumptions in rules $\mathsf{R_{pc1}}$ and $\mathsf{R_{pc2}}$ are not in OWL RL, which does not allow ObjectOneOf in superclasses. Allowing this does not increase reasoning complexity as long as one restricts to exactly one individual in

Table 1. Syntactic simplifications for OWL RL constructs

ObjectMaxCardinality(0 P)	ObjectAllValuesFrom(P owl:Nothing)
ObjectMaxCardinality(0 P C)	ObjectAllValuesFrom(P ObjectComplementOf(C))
ObjectHasValue(P d)	ObjectSomeValuesFrom(P ObjectOneOf(c))
ObjectOneOf($a_1 \ldots a_n$)	ObjectUnionOn(ObjectOneOf(a_1) ... ObjectOneOf(a_n))
EquivalentClasses($C_1 \ldots\ C_n$)	SubClassOf(C_1 C_2), ..., SubClassOf(C_n C_1)
DisjointClasses($C_1 \ldots\ C_n$)	SubClassOf(ObjectIntersectionOf(C_i C_j) owl:Nothing) for all $1 \leq i < j \leq n$
ObjectPropertyDomain(P C)	SubClassOf(ObjectSomeValuesFrom(P owl:Thing) C)
ObjectPropertyRange(P C)	SubClassOf(owl:Thing ObjectAllValuesFrom(P C))
EquivalentObjectProperties($P_1 \ldots\ P_n$)	SubObjectPropertyOf(P_1 P_2), ..., SubObjectPropertyOf(P_n P_1)
InverseObjectProperties(P Q)	SubObjectPropertyOf(P ObjectInverseOf(Q)), SubObjectPropertyOf(Q ObjectInverseOf(P))
SymmetricObjectProperty(P)	SubObjectPropertyOf(P ObjectInverseOf(P))
TransitiveObjectProperty(P)	SubObjectPropertyOf(ObjectPropertyChain(P P) P)
FunctionalObjectProperty(P)	SubClassOf(owl:Thing ObjectMaxCardinality(1 P))
InverseFunctionalObjectProperty(P)	SubClassOf(owl:Thing ObjectMaxCardinality(1 ObjectInverseOf(P)))
AsymmetricObjectProperty(P)	DisjointObjectProperties(P ObjectInverseOf(P))
SameIndividual($a_1 \ldots\ a_n$)	SubClassOf(ObjectOneOf(a_1) ObjectOneOf(a_2)), ..., SubClassOf(ObjectOneOf(a_n) ObjectOneOf(a_1))
NegativeObjectPropertyAssertion(P a b)	SubClassOf(ObjectOneOf(a) ObjectComplementOf(ObjectSomeValuesFrom(P ObjectOneOf(b)))
DifferentIndividuals($a_1 \ldots\ a_n$)	SubClassOf(ObjectOneOf(a_i) ObjectComplementOf(ObjectOneOf(a_j))) for all $1 \leq i < j \leq n$

Table 2. Subclasses (**CL**) and superclasses (**CR**) in syntactically simplified OWL RL

CL ::= **Class** | ObjectIntersectionOf(**CL CL**) | ObjectUnionOf(**CL CL**) |
ObjectOneOf(**Individual**) | ObjectSomeValuesFrom(**Property CL**)
CR ::= **Class** | ObjectIntersectionOf(**CR CR**) | ObjectComplementOf(**CL**) |
ObjectAllValuesFrom(**Property CR**) | ObjectMaxCardinality(1 **Property CL**) |
ObjectSomeValuesFrom(**Property** ObjectOneOf(**Individual**)) | ObjectOneOf(**Individual**)

ObjectOneOf (indeed, this is also done in the tractable OWL EL profile). We will therefore introduce such expressions whenever this simplifies presentation.

Syntactic Sugar. Many OWL features are directly expressible in terms of others. Table 1 lists straightforward syntactic transformations. C, D, E denote class expressions, P, Q object property expressions, and all lower-case letters denote individuals. Below, we can therefore disregard all features on the left of this table. As before, some of the expressions that we use here are not in OWL RL. Besides ObjectOneOf superclasses as discussed above, OWL RL also disallows owl:Thing to be used as a subclass. Again, introducing this does not complicate reasoning. The main motivation for leaving out owl:Thing in the standard is that it may lead to a big number of "uninteresting" entailments, since every individual is an instance of owl:Thing.

In summary, we therefore consider only OWL class inclusion axioms of the form SubClassOf(**CL** **CR**), where **CL** and **CR** are defined as in Table 2.

$$\mathbf{C}_{\mathsf{sco}} \quad \frac{\mathsf{SubClassOf(}\ C\ \ D\ \mathsf{)}\quad \mathsf{SubClassOf(}\ D\ \ E\ \mathsf{)}}{\mathsf{SubClassOf(}\ C\ \ E\ \mathsf{)}}$$

$$\mathbf{C}_{\mathsf{init}} \quad \frac{C \text{ a class expression in the ontology}}{\mathsf{SubClassOf(}\ C\ \ C\ \mathsf{)}\quad \mathsf{SubClassOf(}\ C\ \ \mathsf{owl{:}Thing}\ \mathsf{)}}$$

$$\mathbf{C}_{\mathsf{int}-} \quad \frac{\mathsf{SubClassOf(}\ C\ \ \mathsf{ObjectIntersectionOf(}\ D_1\ D_2\ \mathsf{)}\ \mathsf{)}}{\mathsf{SubClassOf(}\ C\ \ D_1\ \mathsf{)}\quad \mathsf{SubClassOf(}\ C\ \ D_2\ \mathsf{)}}$$

$$\mathbf{C}_{\mathsf{int}+} \quad \frac{\mathsf{SubClassOf(}\ C\ \ D_1\ \mathsf{)}\quad \mathsf{SubClassOf(}\ C\ \ D_2\ \mathsf{)}}{\mathsf{SubClassOf(}\ C\ \ \mathsf{ObjectIntersectionOf(}\ D_1\ D_2\ \mathsf{)}\ \mathsf{)}}$$

$$\mathbf{C}_{\mathsf{com}-} \quad \frac{\mathsf{SubClassOf(}\ C\ \ D\ \mathsf{)}\quad \mathsf{SubClassOf(}\ C\ \ \mathsf{ObjectComplementOf(}D\mathsf{)}\ \mathsf{)}}{\mathsf{SubClassOf(}\ C\ \ \mathsf{owl{:}Nothing}\ \mathsf{)}}$$

$$\mathbf{C}_{\mathsf{uni}+} \quad \frac{\mathsf{SubClassOf(}\ C\ \ D\ \mathsf{)}\quad \text{where } D = D_1 \text{ or } D = D_2}{\mathsf{SubClassOf(}\ C\ \ \mathsf{ObjectUnionOf(}\ D_1\ D_2\ \mathsf{)}\ \mathsf{)}}$$

$$\mathbf{C}_{\mathsf{sa}} \quad \frac{\mathsf{SubClassOf(}\ \mathsf{ObjectOneOf(}c\mathsf{)}\ \ \mathsf{ObjectOneOf(}d\mathsf{)}\ \mathsf{)}}{\mathsf{SubClassOf(}\ \mathsf{ObjectOneOf(}d\mathsf{)}\ \ \mathsf{ObjectOneOf(}c\mathsf{)}\ \mathsf{)}}$$

Fig. 2. OWL RL inference rules for class subsumptions

4 A Rule-Based Classification Calculus for OWL RL

In this section, we specify a class subsumption algorithm for OWL RL, and we introduce conditions under which it is complete. Using the simplifications of the previous section, the only OWL axioms that we need to consider are SubClassOf, SubObjectPropertyOf, DisjointObjectProperties, IrreflexiveObjectProperty, and HasKey. Remaining expressive features are those given in Table 2 (for class expressions) and ObjectPropertyChain (for property inclusions).

Figures 2 and 3 specify a rule system for deriving class subsumptions, where we use the same notation for rules as above. The rules in Fig. 2 apply to class subsumptions and, using the correspondences of Fig. 1, also to assertions. In contrast, the rules in Fig. 3 are only applicable to specific assertions and are not generalised to other subsumptions. For example, rule $\mathbf{P}_{\mathsf{inv}-}$ is sound, but the following generalisation to class inclusions would of course be wrong:

$$\frac{\mathsf{SubClassOf(}\ C\ \ \mathsf{ObjectSomeValuesFrom(}\ \mathsf{ObjectInverseOf(}P\mathsf{)}\ D\ \mathsf{)}\ \mathsf{)}}{\mathsf{SubClassOf(}\ D\ \ \mathsf{ObjectSomeValuesFrom(}\ P\ C\ \mathsf{)}\ \mathsf{)}}$$

As an additional condition for applying rules, we require that all class and property expressions in the conclusion also occur in the ontology. This is a restriction only for the rules $\mathbf{C}_{\mathsf{int}+}$, $\mathbf{C}_{\mathsf{uni}+}$, $\mathbf{P}_{\mathsf{svf}+}$, and $\mathbf{P}_{\mathsf{inv}+}$, since they are the only rules that derive expressions that are not mentioned in their premise.

For an OWL ontology $\mathcal{O}$, we say that a class subsumption SubClassOf(C D) is *inferred* by the rule system if one of the following axioms is derived by applying the rules exhaustively to $\mathcal{O}$:

- SubClassOf(C D),
- SubClassOf(C owl:Nothing), or
- ClassAssertion(owl:Nothing c) for some individual c.

$$\mathsf{P}_{\mathsf{avf}-}\quad \frac{\mathsf{ObjectPropertyAssertion}(\ P\ c\ d\ )\quad \mathsf{ClassAssertion}(\ \mathsf{ObjectAllValuesFrom}(\ P\ E\ )\ c\ )}{\mathsf{ClassAssertion}(\ E\ d\ )}$$

$$\mathsf{P}_{\mathsf{svf}+}\quad \frac{\mathsf{ObjectPropertyAssertion}(\ P\ c\ d\ )\quad \mathsf{ClassAssertion}(\ E\ d\ )}{\mathsf{ClassAssertion}(\ \mathsf{ObjectSomeValuesFrom}(\ P\ E\ )\ c\ )}$$

$$\mathsf{P}_{\mathsf{inv}-}\quad \frac{\mathsf{ObjectPropertyAssertion}(\ \mathsf{ObjectInverseOf}(P)\ c\ d\ )}{\mathsf{ObjectPropertyAssertion}(\ P\ d\ c\ )}$$

$$\mathsf{P}_{\mathsf{inv}+}\quad \frac{\mathsf{ObjectPropertyAssertion}(\ P\ d\ c\ )}{\mathsf{ObjectPropertyAssertion}(\ \mathsf{ObjectInverseOf}(P)\ c\ d\ )}$$

$$\mathsf{P}_{\mathsf{spo}}\quad \frac{\mathsf{ObjectPropertyAssertion}(\ P\ c\ d\ )\quad \mathsf{SubObjectPropertyOf}(\ P\ Q\ )}{\mathsf{ObjectPropertyAssertion}(\ Q\ c\ d\ )}$$

$$\mathsf{P}_{\mathsf{spc}}\quad \frac{\begin{array}{c}\mathsf{SubObjectPropertyOf}(\ \mathsf{ObjectPropertyChain}(\ P\ Q\ )\ R\ )\\ \mathsf{ObjectPropertyAssertion}(\ P\ c\ d\ )\quad \mathsf{ObjectPropertyAssertion}(\ Q\ d\ e\ )\end{array}}{\mathsf{ObjectPropertyAssertion}(\ R\ c\ e\ )}$$

$$\mathsf{P}_{\mathsf{dp}}\quad \frac{\begin{array}{c}\mathsf{DisjointObjectProperties}(\ P\ Q\ )\\ \mathsf{ObjectPropertyAssertion}(\ P\ c\ d\ )\quad \mathsf{ObjectPropertyAssertion}(\ Q\ c\ d\ )\end{array}}{\mathsf{ClassAssertion}(\ \mathsf{owl{:}Nothing}\ c\ )}$$

$$\mathsf{P}_{\mathsf{ip}}\quad \frac{\mathsf{ObjectPropertyAssertion}(\ P\ c\ c\ )\quad \mathsf{IrreflexiveObjectProperty}(P)}{\mathsf{ClassAssertion}(\ \mathsf{owl{:}Nothing}\ c\ )}$$

$$\mathsf{P}_{\mathsf{key}}\quad \frac{\begin{array}{c}\mathsf{HasKey}(\ E\ (P_1 \ldots P_n)\ ()\ )\\ \mathsf{ClassAssertion}(\ E\ c\ )\quad \mathsf{ClassAssertion}(\ E\ d\ )\\ \mathsf{ObjectPropertyAssertion}(\ P_1\ c\ e_1\ )\ \ldots\ \mathsf{ObjectPropertyAssertion}(\ P_n\ c\ e_n\ )\\ \mathsf{ObjectPropertyAssertion}(\ P_1\ d\ e_1\ )\ \ldots\ \mathsf{ObjectPropertyAssertion}(\ P_n\ d\ e_n\ )\end{array}}{\mathsf{SubClassOf}(\ \mathsf{ObjectOneOf}(c)\ \mathsf{ObjectOneOf}(d)\ )}$$

$$\mathsf{P}_{\mathsf{fun}}\quad \frac{\begin{array}{c}\mathsf{ClassAssertion}(\ \mathsf{ObjectMaxCardinality}(\ 1\ P\ D\ )\ c\ )\\ \mathsf{ObjectPropertyAssertion}(\ P\ c\ e_1\ )\quad \mathsf{ObjectPropertyAssertion}(\ P\ c\ e_2\ )\\ \mathsf{ClassAssertion}(\ D\ e_1\ )\quad \mathsf{ClassAssertion}(\ D\ e_2\ )\end{array}}{\mathsf{SubClassOf}(\ \mathsf{ObjectOneOf}(e_1)\ \mathsf{ObjectOneOf}(e_2)\ )}$$

Fig. 3. OWL RL inference rules that are specific to property assertions

The first condition corresponds to a direct derivation, the second captures the case that C is inconsistent (necessarily empty), and the third case occurs whenever $\mathcal{O}$ is inconsistent. The inference rules in [16] use a special conclusion *false* to encode ontology inconsistency, but this just a minor difference.

Theorem 1. *The rule system is sound, that is, if a class subsumption A is inferred from $\mathcal{O}$, then $\mathcal{O}$ entails A under the Direct Semantics and under the RDF-based Semantics of OWL.*

However, the rule system is not complete. The following examples illustrate two interesting cases that are not covered.

Example 3. We consider an ontology of four axioms:

InstanceOf(:D :d) (7)

SubClassOf(:D :C) (8)

SubClassOf(:C ObjectHasValue(:P :a)) (9)

InstanceOf(ObjectMaxCardinality(1 ObjectInverseOf(:P) owl:Thing) :a) (10)

From this, the axiom SubClassOf(:C :D) follows. Indeed, (9) and (10) together imply that :C can have at most one instance; by (7) and (8), this instance is :d and thus contained in :D. However, this is not entailed by our rule system. Axioms (9) and (10) can be represented as follows:

SubClassOf(:C ObjectSomeValuesFrom(:P ObjectOneOf(:a))) (11)

InstanceOf(ObjectAllValuesFrom(ObjectInverseOf(:P) ObjectOneOf(:e)) :a) (12)

where :e is an auxiliary individual. Using $\mathsf{C_{sco}}$ and the rules of Fig. 1, we can derive ObjectPropertyAssertion(:P :d :a) from (7), (8), and (11). By applying rule $\mathsf{P_{inv+}}$, we obtain ObjectPropertyAssertion(ObjectInverseOf(:P) :a :d). Together with (12), $\mathsf{P_{avf-}}$ implies ClassAssertion(ObjectOneOf(:e) :d). The same could be derived for any other instance :d' of :C, showing that all such instances must be equal, and instances of :D. However, we cannot derive SubClassOf(:C :D).

Example 4. Consider the ontology of the following axioms:

SubClassOf(:C ObjectHasValue(:Q :b)) (13)

ObjectPropertyRange(:Q :D) (14)

ObjectPropertyDomain(:Q :D) (15)

SubClassOf(:D ObjectHasValue(:P :a)) (16)

SubObjectPropertyOf(ObjectPropertyChain(:P ObjectInverseOf(:P)) :R) (17)

SubClassOf(ObjectSomeValuesFrom(:R owl:Thing) :E) (18)

These axioms imply SubClassOf(:C :E). Indeed, axioms (16) and (17) together imply that every pair of instances of :D is connected by property :R. Axioms (14) and (15) in turn imply that :Q only connects instances that are in :D. Thus, we find that :Q is a subproperty of :P, which is an interesting inference in its own right. Combining this with (13) and (18), we obtain the claimed entailment SubClassOf(:C :E). Again, this is not inferred by our inference rules.

Examples 3 and 4 illustrate two very different forms of semantic interactions that lead to entailments not inferred by our rule system. It is interesting to note, however, that ObjectHasValue plays a crucial role in both cases. Indeed, we find that this feature is involved in every situation where an entailment is missed:

Theorem 2. *If $\mathcal{O}$ is an OWL RL ontology that does not contain* ObjectHasValue *in superclasses, then the rule system is complete for $\mathcal{O}$. More precisely, let $\mathcal{O}'$ be the syntactic simplification of $\mathcal{O}$ as discussed above. If* :A *and* :B *are class names and $\mathcal{O}$ entails* SubClassOf(:A :B) *under Direct Semantics, then* SubClassOf(:A :B) *is inferred from $\mathcal{O}'$ by the rule system.*

Even in this case, we need to take assertions into account during reasoning, since they might make the ontology inconsistent. For consistent ontologies, however, the rule system can be simplified further.

Theorem 3. *For a consistent OWL RL ontology $\mathcal{O}$ that does not contain* ObjectHasValue *in superclasses, all entailed subsumptions between class names can be computed using the rules of Fig. 2 only, without taking assertions into account.*

More precisely, let $\mathcal{O}_t \subseteq \mathcal{O}$ be the set of all axioms in $\mathcal{O}$ that do not use ClassAssertion*,* ObjectPropertyAssertion*,* SameIndividual*,* DifferentIndividuals*, or* NegativeObjectPropertyAssertion*. Let $\mathcal{O}'_t$ be the syntactic simplification of $\mathcal{O}_t$. If $\mathcal{O}$ entails* SubClassOf(:A :B) *under Direct Semantics, and* :A *and* :B *are class names, then* SubClassOf(:A :B) *or* SubClassOf(:A owl:Nothing) *is derived from $\mathcal{O}'_t$ by the rules of Fig. 2.*

This tells us that the computation of class subsumptions in OWL RL is indeed very simple for consistent ontologies without ObjectHasValue in superclasses. Provided that this situation can be assumed, it would therefore be feasible to pre-compute class subsumptions without taking assertions into account. In data-centric ontologies, this can lead to a much smaller set of axioms. In addition, the set of rules that need to be applied becomes relatively small as well.

5 RDF-Based Rule Systems

We have formulated rules above using the Functional-Style Syntax of OWL. In this section, we explain how these results transfer to RDF-based rules in the style of the W3C specification [16], which act on the RDF serialisation of OWL [18]. This also prepares the ground for discussing the limitations of such rules in the next section.

Definition 1. *An* RDF-based rule *is a first-order implication of the form*

$$T(s_1, t_1, u_1) \wedge \ldots \wedge T(s_n, t_n, u_n) \rightarrow T(s, t, u)$$

where $s_{(i)}$, $t_{(i)}$, and $u_{(i)}$ are RDF terms (i.e., IRIs, literals, or blank nodes) or first-order logic variables. All variables are assumed to be universally quantified.

An (RDF-based) rule system *is a finite set $\mathcal{R}$ of RDF-based rules. $\mathcal{R}$ is applied to an RDF graph by considering RDF triples $\langle s\ p\ o \rangle$ as facts $T(s, p, o)$. A rule system $\mathcal{R}$ is applied to an OWL ontology $\mathcal{O}$ by applying it to the RDF serialisation of $\mathcal{O}$ [18].*

A rule system $\mathcal{R}$ is a sound and complete classification system *for a class $\mathcal{C}$ of ontologies if, for every ontology $\mathcal{O}$ in $\mathcal{C}$ and all class names* :A *and* :B*:*

SubClassOf(:A :B) *is entailed by $\mathcal{O}$*
if and only if
$\mathcal{R}$ infers $T($:A, rdfs:subClassOf, :B*$)$ from $\mathcal{O}$.*

The main features of RDF-based rule systems are thus as follows:

– Finiteness: the set of rules is finite
– Monotonicity: the larger the set of (derived) facts, the larger the number of rules that can be applied
– No value invention: applying rules does not introduce new terms
– Uniformity: the applicability of rules does not depend on the IRIs of the entities it is applied to, but only on the statements that these IRIs occur in
– Triple-based: the only relation symbol used in inferences is the ternary T

The W3C OWL RL inference rules do not constitute a rule system in the sense of Definition 1, since they are not finite in number. The reason is that the rules support list-based OWL features for lists of arbitrary length, leading to an infinite number of possible patterns. A rule system that is not restricted to be finite can trivially solve all reasoning tasks: for every ontology $\mathcal{O}$ for which we require an inference $T(s, p, o)$, we can add a rule $T_{\mathcal{O}} \to T(s, p, o)$, where $T_{\mathcal{O}}$ is the triple pattern that corresponds to the RDF serialisation of $\mathcal{O}$. It is also clear, however, that the infinite W3C rule system does not use this potential power.

Theorem 4. *The rules of Section 4 give rise to an RDF-based sound and complete classification system for (the syntactic simplification of) OWL RL ontologies without* ObjectHasValue *in superclasses.*

This result is based on the rule system from Section 4, which we already know to be sound and complete. These rules can be easily expressed in the RDF serialisation of OWL based on the T predicate. For example, $\mathsf{C}_{\mathsf{int}+}$ can be written as follows, where question marks denote variables as in [16]:

$$\begin{aligned} T(?x, \text{rdfs:subClassOf}, ?y_1) \wedge T(?x, \text{rdfs:subClassOf}, ?y_2) \wedge \\ T(?c, \text{owl:intersectionOf}, ?l) \wedge T(?l, \text{rdf:first}, ?y_1) \wedge T(?l, \text{rdf:rest}, ?l') \wedge \\ T(?l', \text{rdf:first}, ?y_2) \wedge T(?l', \text{rdf:rest}, \text{rdf:nil}) \\ \to T(?x, \text{rdfs:subClassOf}, ?c) \end{aligned}$$

This is the rule that is mainly missing from [16]. Note how the check for the existence of the expression ObjectIntersectionOf(D_1 D_2) that is required for applying $\mathsf{C}_{\mathsf{int}+}$ is performed naturally as part of the rule application. This does not work for the rules in Fig. 1, which do not require existence of all class expressions in the consequence. Either they are created as part of the syntactic simplification stage, or the remaining rules need to be extended to allow for multiple alternative forms of each premise. The latter approach is taken in the W3C specification.

6 Rule-Based Classification of Unrestricted OWL RL

So far, we have seen that the (RDF-based) rule systems for OWL RL can be extended to obtain a sound and complete approach for computing class subsumptions, as long as we disallow ObjectHasValue in superclasses. The natural question is whether this can be extended to arbitrary OWL RL ontologies. The answer is *no*, as we will show in this section.

This negative result is caused by the use of RDF as a basis for deduction rules. Without this restriction, it is not hard to find rule-based classification systems that are sound and complete, though not necessarily efficient. Namely, one can always check SubClassOf(C D) by adding an axiom InstanceOf(C e) for a new individual e, and checking whether InstanceOf(D e) using the existing rule-based approaches for instance retrieval. The problem is that this executes a single check, based on a modification of the ontology, rather than computing all class subsumptions at once. Tests for different classes may lead to different inferences. To address this, we can augment each inferred axiom with the test class C for which we have assumed InstanceOf(C e) (the IRI of e is immaterial and does not need to be recorded). The rules are restricted to the case that all premises can be derived under the same assumption, and computation can proceed concurrently without interactions between independent assumptions. Similar solutions and various optimisations have been suggested and implemented for OWL EL [14,10]. Unfortunately, however, it is not obvious how to represent such an extended form of inferences in RDF without introducing new entities or blank nodes.

The main result of this section is that this problem is not just due to an overly naive approach for extending the rules, but rather an inherent limitation of RDF-based rules:

Theorem 5. *There is no RDF-based sound and complete classification system for OWL RL.*

This is a very general result, since it makes a statement about *every* conceivable RDF-based rule system. A technique for proving such results has been developed in the context of rule-based reasoning for OWL EL [14]. Recalling this in full detail is beyond the scope of this paper. Instead, we extract the main insights on the level of ontological reasoning (Lemma 1), and present the crucial steps for applying this approach to our scenario.

The argumentation is based on the analysis of derivations in rule systems, which can be represented as *proof trees*. The key observation is that every proof tree of a rule system can be used to construct further proof trees by selectively renaming constants (i.e., IRIs). This renaming leads to proof trees that derive the same conclusion, applying rules in the same order, but based on a renamed input ontology. The renaming may not be uniform, that is, multiple occurrences of the same constant might be renamed differently. The proof uses the fact that such renaming may destroy entailments. Ontologies that are particularly sensitive in this respect are called *critical*:

Definition 2. *A* renaming *of an ontology $\mathcal{O}$ is an ontology $\mathcal{O}'$ that is obtained from $\mathcal{O}$ by replacing occurrences of entity names by fresh entity names, where neither the old nor the new entities have a special semantics in OWL. A renaming is* uniform *if all occurrences of one entity have been replaced in the same way; otherwise it is* non-uniform.

An ontology $\mathcal{O}$ is critical *for an axiom A if A is entailed by $\mathcal{O}$, and A is not entailed by any non-uniform renaming of $\mathcal{O}$.*

Roughly speaking, an ontology is critical if all of its axioms are really needed for obtaining the desired conclusion. The next result explains the significance of critical ontologies $\mathcal{O}$. It states that, given a sound and complete classification system, it must be possible to decompose $\mathcal{O}$ into sets $\mathcal{O}_{\text{split}}$ and $\mathcal{O} \setminus \mathcal{O}_{\text{split}}$. The requirement is that both sets "touch" in at most 3 axioms, which reflects the arity of RDF triples. By finding a critical ontology for which this is not possible, we can show that there is no sound and complete classification system.

Lemma 1. *Suppose that $\mathcal{R}$ is a sound and complete classification system with at most ℓ atoms $T(s,t,u)$ in the premise of any rule, and consider an ontology $\mathcal{O}$ that is critical for an axiom* SubClassOf(C D) *with class names C and D.*

For every $\mathcal{O}' \subseteq \mathcal{O}$ with $|\mathcal{O}'| > 3(\ell+1)$, there is $\mathcal{O}_{split} \subseteq \mathcal{O}$ such that:

– $|\mathcal{O}' \cap \mathcal{O}_{split}| \geq 4$,
– $|\mathcal{O}' \cap (\mathcal{O} \setminus \mathcal{O}_{split})| \geq 4$,
– *at most* 3 *axioms in $\mathcal{O}_{split}$ share any vocabulary symbols with $\mathcal{O} \setminus \mathcal{O}_{split}$.*

This result is obtained as a summary of various insights from [14]. Lemma 1 requires critical ontologies to contain at least $3(\ell+1)$ axioms. Since ℓ depends on the rule system we consider, we require critical OWL RL ontologies of arbitrary size. The following definition provides this crucial ingredient.

Definition 3. *For every natural number $k \geq 0$, we define an ontology $\mathcal{O}_k$ as follows. We consider class names A, B, D_i ($i = 0, \ldots, k+1$), property names V, W, P_i ($i = 0, \ldots, k$), Q_i ($i = 0, \ldots, k+1$), and individual names a, b, c, d_i ($i = 1, \ldots, k+1$). The ontology $\mathcal{O}_k$ consists of the following axioms:*

SubClassOf(D_i ObjectHasValue(P_i d_{i+1}) $\quad 0 \leq i \leq k \quad$ (19)

SubClassOf(D_i ObjectAllValuesFrom(P_i D_{i+1}) $\quad 0 \leq i \leq k \quad$ (20)

SubClassOf(D_0 ObjectHasValue(W a)) $\quad$ (21)

SubClassOf(D_0 ObjectAllValuesFrom(W A)) $\quad$ (22)

SubClassOf(A C) $\quad$ (23)

SubClassOf(C ObjectHasValue(Q_{k+1} b)) $\quad$ (24)

SubClassOf(A ObjectHasValue(V c)) $\quad$ (25)

SubClassOf(D_{k+1} ObjectHasValue(V c)) $\quad$ (26)

InverseFunctionalObjectProperty(V) $\quad$ (27)

SubObjectPropertyOf(ObjectPropertyChain(P_i Q_{i+1}) Q_i) $\quad 0 \leq i \leq k \quad$ (28)

SubClassOf(ObjectHasValue(Q_0 b) B) $\quad$ (29)

Lemma 2. *For every $k \geq 0$, $\mathcal{O}_k$ entails* SubClassOf(D_0 B)*.*

This can be seen as follows: If D_0 does not contain any instances, then the statement clearly holds. If D_0 contains some instance o, then it is the start of a

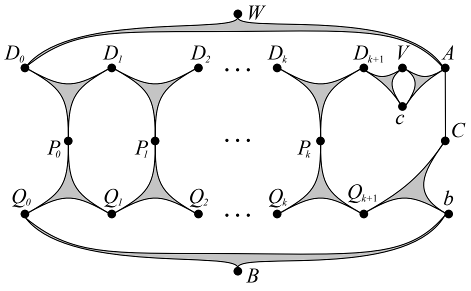


Fig. 4. Illustration of syntactic dependencies in $\mathcal{O}_k$

property chain $P_0, \ldots, P_k$ (through individuals $d_1, \ldots, d_{k+1}$) due to (19) and (20). In particular, d_k is an instance of D_{k+1}, hence, by (26), d_k has a V successor c. Similarly, by (21) and (22), a is an instance of A. By (23), (24), and (25), a thus has a Q_{k+1} successor b and a V successor c. Since V is inverse functional (27), a must therefore be equal to d_k, so d_k has a Q_{k+1} successor b. Applying axioms (28) to the chain of d_i elements, we find that d_i has the Q_i successor b for all $1 \leq i \leq k$. Accordingly, the instance o of D_0 has Q_0 successor b. By (29), o thus is an instance of B. Since this reasoning applies to every instance o of D_0, we find that SubClassOf(D_0 B) as claimed.

It is not hard to see that this entailment is no longer valid if any two occurrences of symbols within $\mathcal{O}_k$ are renamed in a non-uniform way:

Lemma 3. *For every $k \geq 0$, $\mathcal{O}_k$ is critical for* SubClassOf(D_0 B)*.*

To complete the proof of Theorem 5 we thus need to argue that there are critical ontologies that cannot be split as in Lemma 1.

Lemma 4. *Consider $\mathcal{O}_k$ and let $\mathcal{O}'$ be the subset of all axioms* (20)*. For every $\mathcal{O}_{split} \subseteq \mathcal{O}$ such that $|\mathcal{O}_{split} \cap \mathcal{O}'| \geq 4$ and $|\mathcal{O}' \setminus \mathcal{O}_{split}| \geq 4$, there are at least 4 axioms in $\mathcal{O}_{split}$ that share vocabulary symbols with $\mathcal{O} \setminus \mathcal{O}_{split}$.*

Proof. We can illustrate the syntactic dependencies in an ontology by means of a (hyper)graph where each axiom is an edge between all entities that it refers to. Figure 4 shows part of the according graph for $\mathcal{O}_k$, showing only axioms (20), (22)–(26), (28), and (29). A subset of these axioms thus corresponds to a subset of edges, and the shared vocabulary symbols are shared nodes.

The assumptions on $\mathcal{O}_{\text{split}}$ require that $\mathcal{O}_{\text{split}}$ contains at least 4 of the axioms (20) (upper row in Fig. 4), and also misses at least 4 of the axioms (20). Using the illustration in Fig. 4, it is not hard to see that this requires $\mathcal{O}_{\text{split}}$ to share signature symbols with at least 4 axioms not in $\mathcal{O}_{\text{split}}$. □

Thus, for any rule system $\mathcal{R}$ with at most ℓ atoms in rule premises, the ontology $\mathcal{O}_k$ for $k = 3(\ell + 1)$ is critical but does not satisfy the conditions of Lemma 1. Thus, $\mathcal{R}$ cannot be a sound and complete classification system.

7 Conclusion

From a practical perspective, the main contribution of this work is to clarify the problems of OWL RL classification, and to propose rule systems for solving this task in relevant cases. The rules that we proposed produce sound conclusions on arbitrary OWL ontologies, under either of the two semantics of OWL. If the input is an OWL RL ontology where ObjectHasValue is not used in superclasses, the rule system is also guaranteed to be complete.

Our findings have also been brought to the attention of the OWL Working Group, which is preparing an editorial update of the OWL 2 specification at the time of this writing. This new version will correct the complexity claims about OWL RL. Extending the inference rules to be complete for computing class subsumptions in ontologies without ObjectHasValue, however, is beyond the scope of this editorial update. OWL RL tools can achieve completeness in this sense by adding, in essence, the one additional rule given after Theorem 4 (generalised to conjunctions of arbitrary arity). This does not affect official conformance.

Interestingly, ObjectHasValue in superclasses is the one OWL RL feature that complicates schema reasoning the most. This contrasts with OWL EL, where such expressions are as easy to handle as assertions [10]. The reason is that inverse properties, ObjectAllValuesFrom, and ObjectMaxCardinality, all of which allow for some complicated interactions with ObjectHasValue, are not in OWL EL.

Another interesting insight of this work is that there are practical problems in OWL RL reasoning that RDF-based rules are too inexpressive to solve. This limitation is due to the triple-based representation of RDF, which could be overcome by allowing predicates of higher arities as in Datalog [1] or RIF [11]. For keeping closer to features supported in RDF databases, it might be possible to use *quads* or *named graphs* for expressing 4-ary predicates, but it is doubtful if this would be an adequate use of these features. On the other hand, 4-ary relations are only needed as intermediate results during reasoning, so individual systems can implement solutions without referring to any language standard.

Another approach is to allow rules with value creation (*blank nodes* in rule heads) to encode n-ary relationships by introducing auxiliary entities. Value invention is problematic in general, as it can lead to non-termination and undecidability. Many works have studied conditions that ensure termination of bottom-up reasoning in the presence of value creation – see [3] for a recent overview – but it is unclear if any of these conditions would apply in our case.

Acknowledgements. The research reported herein was supported by the EPSRC projects ConDOR and ExODA.

References

1. Abiteboul, S., Hull, R., Vianu, V.: Foundations of Databases. Addison Wesley (1994)
2. Bishop, B., Kiryakov, A., Ognyanoff, D., Peikov, I., Tashev, Z., Velkov, R.: OWLIM: a family of scalable semantic repositories. Semantic Web Journal 2(1), 33–42 (2011)

3. 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. 13th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR 2012), pp. 243–253. AAAI Press (2012)
4. Erling, O.: Virtuoso, a hybrid RDBMS/graph column store. IEEE Data Eng. Bull. 35(1), 3–8 (2012)
5. Franz Inc.: AllegroGraph RDFStore: Web 3.0's Database (2012), http://www.franz.com/agraph/allegrograph/ (accessed April 2012)
6. Hitzler, P., Krötzsch, M., Parsia, B., Patel-Schneider, P.F., Rudolph, S. (eds.): OWL 2 Web Ontology Language: Primer. W3C Recommendation (October 27, 2009), http://www.w3.org/TR/owl2-primer/
7. Hogan, A., Harth, A., Polleres, A.: Scalable authoritative OWL reasoning for the Web. Int. J. of Semantic Web Inf. Syst. 5(2), 49–90 (2009)
8. Hogan, A., Pan, J.Z., Polleres, A., Decker, S.: SAOR: Template Rule Optimisations for Distributed Reasoning over 1 Billion Linked Data Triples. 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. 337–353. Springer, Heidelberg (2010)
9. ter Horst, H.J.: Completeness, decidability and complexity of entailment for RDF Schema and a semantic extension involving the OWL vocabulary. J. of Web Semantics 3(2-3), 79–115 (2005)
10. Kazakov, Y., Krötzsch, M., Simančík, F.: Practical reasoning with nominals in the $\mathcal{EL}$ family of description logics. In: Proc. 13th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR 2012), pp. 264–274. AAAI Press (2012)
11. Kifer, M., Boley, H. (eds.): RIF Overview. W3C Working Group Note (June 22, 2010), http://www.w3.org/TR/rif-overview/
12. 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)
13. Kotoulas, S., Oren, E., van Harmelen, F.: Mind the data skew: distributed inferencing by speeddating in elastic regions. In: Proc. 19th Int. Conf. on World Wide Web (WWW 2010), pp. 531–540. ACM (2010)
14. Krötzsch, M.: Efficient rule-based inferencing for OWL EL. In: Walsh, T. (ed.) Proc. 22nd Int. Joint Conf. on Artificial Intelligence (IJCAI 2011), pp. 2668–2673. AAAI Press/IJCAI (2011)
15. Krötzsch, M.: The (not so) easy task of computing class subsumptions in OWL RL. Tech. rep., University of Oxford (2012), http://korrekt.org/page/OWLRL2012
16. 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/
17. Motik, B., Patel-Schneider, P.F., Parsia, B. (eds.): OWL 2 Web Ontology Language: Structural Specification and Functional-Style Syntax. W3C Recommendation (October 27, 2009), http://www.w3.org/TR/owl2-syntax/
18. Patel-Schneider, P.F., Motik, B. (eds.): OWL 2 Web Ontology Language: Mapping to RDF Graphs. W3C Recommendation (October 27, 2009), http://www.w3.org/TR/owl2-mapping-to-rdf/
19. Schenk, S., Staab, S.: Networked graphs: a declarative mechanism for SPARQL rules, SPARQL views and RDF data integration on the Web. In: Proc. 17th Int. Conf. on World Wide Web (WWW 2008), pp. 585–594. ACM (2008)
20. Urbani, J., Kotoulas, S., Maassen, J., van Harmelen, F., Bal, H.: WebPIE: a Web-scale parallel inference engine using MapReduce. J. of Web Semantics 10, 59–75 (2012)

Strabon: A Semantic Geospatial DBMS*

Kostis Kyzirakos, Manos Karpathiotakis, and Manolis Koubarakis

National and Kapodistrian University of Athens, Greece
{kkyzir,mk,koubarak}@di.uoa.gr

Abstract. We present Strabon, a new RDF store that supports the state of the art semantic geospatial query languages stSPARQL and GeoSPARQL. To illustrate the expressive power offered by these query languages and their implementation in Strabon, we concentrate on the new version of the data model stRDF and the query language stSPARQL that we have developed ourselves. Like GeoSPARQL, these new versions use OGC standards to represent geometries where the original versions used linear constraints. We study the performance of Strabon experimentally and show that it scales to very large data volumes and performs, most of the times, better than all other geospatial RDF stores it has been compared with.

1 Introduction

The Web of data has recently started being populated with geospatial data. A representative example of this trend is project LinkedGeoData where OpenStreetMap data is made available as RDF and queried using the declarative query language SPARQL. Using the same technologies, Ordnance Survey makes available various geospatial datasets from the United Kingdom.

The availability of geospatial data in the linked data "cloud" has motivated research on geospatial extensions of SPARQL [7,9,13]. These works have formed the basis for GeoSPARQL, a proposal for an Open Geospatial Consortium (OGC) standard which is currently at the "candidate standard" stage [1]. In addition, a number of papers have explored implementation issues for such languages [2,3]. In this paper we present our recent achievements in both of these research directions and make the following technical contributions.

We describe a new version of the data model stRDF and the query language stSPARQL, originally presented in [9], for representing and querying geospatial data that change over time. In the new version of stRDF, we use the widely adopted OGC standards Well Known Text (WKT) and Geography Markup Language (GML) to represent geospatial data as literals of datatype `strdf:geometry` (Section 2). The new version of stSPARQL is an extension of SPARQL 1.1 which, among other features, offers functions from the OGC standard "OpenGIS Simple Feature Access for SQL" for the manipulation of spatial

* This work was supported in part by the European Commission project TELEIOS (http://www.earthobservatory.eu/)

literals and support for multiple coordinate reference systems (Section 3). The new version of stSPARQL and GeoSPARQL have been developed independently at about the same time. We discuss in detail their many similarities and few differences and compare their expressive power.

We present the system Strabon[1], an open-source semantic geospatial DBMS that can be used to store linked geospatial data expressed in stRDF and query them using stSPARQL. Strabon can also store data expressed in RDF using the vocabularies and encodings proposed by GeoSPARQL and query this data using the subset of GeoSPARQL that is closer to stSPARQL (the union of the GeoSPARQL core, the geometry extension and the geometry topology extension [1]). Strabon extends the RDF store Sesame, allowing it to manage both thematic and spatial RDF data stored in PostGIS. In this way, Strabon exposes a variety of features similar to those offered by geospatial DBMS that make it one of the richest RDF store with geospatial support available today.

Finally, we perform an extensive evaluation of Strabon using large data volumes. We use a real-world workload based on available geospatial linked datasets and a workload based on a synthetic dataset. Strabon can scale up to 500 million triples and answer complex stSPARQL queries involving the whole range of constructs offered by the language. We present our findings in Section 5, including a comparison between Strabon on top of PostgreSQL and a proprietary DBMS, the implementation presented in [3], the system Parliament [2], and a baseline implementation. Thus, we are the first to provide a systematic evaluation of RDF stores supporting languages like stSPARQL and GeoSPARQL and pointing out directions for future research in this area.

2 A New Version of stRDF

In this section we present a new version of the data model stRDF that was initially presented in [9]. The presentation of the new versions of stRDF and stSPARQL (Section 3) is brief since the new versions are based on the initial ones published in [9]. In [9] we followed the ideas of constraint databases [12] and chose to represent spatial and temporal data as quantifier-free formulas in the first-order logic of linear constraints. These formulas define subsets of $\mathbb{Q}^k$ called *semi-linear point sets* in the constraint database literature. In the original version of stRDF, we introduced the data type `strdf:SemiLinearPointSet` for modeling geometries. The values of this datatype are typed literals (called *spatial* literals) that encode geometries using Boolean combinations of linear constraints in $\mathbb{Q}^2$. For example,

$$(x \geq 0 \land y \geq 0 \land x + y \leq 1) \lor (x \leq 0 \land y \leq 0 \land x + y \geq -1)$$

is such a literal encoding the union of two polygons in $\mathbb{Q}^2$. In [9] we also allow the representation of *valid times of triples* using the proposal of [6]. Valid times are again represented using order constraints over the time structure $\mathbb{Q}$. In the rest of this paper we omit the temporal dimension of stRDF from our discussion and concentrate on the geospatial dimension only.

[1] `http://www.strabon.di.uoa.gr`

Although our original approach in [9] results in a theoretically elegant framework, none of the application domains we worked with had geospatial data represented using constraints. Today's GIS practitioners represent geospatial data using OGC standards such as WKT and GML. Thus, in the new version of stRDF and stSPARQL, which has been used in EU projects SemsorGrid4Env and TELEIOS, the linear constraint representation of spatial data was dropped in favour of OGC standards. As we demonstrate below, introducing OGC standards in stRDF and stSPARQL has been achieved easily without changing anything from the basic design choices of the data model and the query language.

In the new version of stRDF, the datatypes `strdf:WKT` and `strdf:GML` are introduced to represent geometries serialized using the OGC standards WKT and GML. WKT is a widely accepted OGC standard[2] and can be used for representing geometries, coordinate reference systems and transformations between coordinate reference systems. A *coordinate system* is a set of mathematical rules for specifying how coordinates are to be assigned to points. A *coordinate reference system* (CRS) is a coordinate system that is related to an object (e.g., the Earth, a planar projection of the Earth) through a so-called *datum* which specifies its origin, scale, and orientation. Geometries in WKT are restricted to 0-, 1- and 2-dimensional geometries that exist in $\mathbb{R}^2$, $\mathbb{R}^3$ or $\mathbb{R}^4$. Geometries that exist in $\mathbb{R}^2$ consist of points with coordinates x and y, e.g., `POINT(1,2)`. Geometries that exist in $\mathbb{R}^3$ consist of points with coordinates x, y and z or x, y and m where m is a measurement. Geometries that exist in $\mathbb{R}^4$ consist of points with coordinates x, y, z and m. The WKT specification defines syntax for representing the following classes of geometries: points, line segments, polygons, triangles, triangulated irregular networks and collections of points, line segments and polygons. The interpretation of the coordinates of a geometry depends on the CRS that is associated with it.

GML is an OGC standard[3] that defines an XML grammar for modeling, exchanging and storing geographic information such as coordinate reference systems, geometries and units of measurement. The GML Simple Features specification (GML-SF) is a profile of GML that deals only with a subset of GML and describes geometries similar to the one defined by WKT.

Given the OGC specification for WKT, the datatype `strdf:WKT`[4] is defined as follows. The lexical space of this datatype includes finite-length sequences of characters that can be produced from the WKT grammar defined in the WKT specification, optionally followed by a semicolon and a URI that identifies the corresponding CRS. The default case is considered to be the WGS84 coordinate reference system. The value space is the set of geometry values defined in the WKT specification. These values are a subset of the union of the powersets of $\mathbb{R}^2$ and $\mathbb{R}^3$. The lexical and value space for `strdf:GML` are defined similarly. The datatype `strdf:geometry` is also introduced to represent the serialization of a geometry independently of the serialization standard used. The datatype

[2] http://portal.opengeospatial.org/files/?artifact_id=25355

[3] http://portal.opengeospatial.org/files/?artifact_id=39853

[4] http://strdf.di.uoa.gr/ontology

`strdf:geometry` is the union of the datatypes `strdf:WKT` and `strdf:GML`, and appropriate relationships hold for its lexical and value spaces.

Both the original [9] and the new version of stRDF presented in this paper impose minimal new requirements to Semantic Web developers that want to represent spatial objects with stRDF; all they have to do is utilize a new literal datatype. These datatypes (`strdf:WKT`, `strdf:GML` and `strdf:geometry`) can be used in the definition of geospatial ontologies needed in applications, e.g., ontologies similar to the ones defined in [13].

In the examples of this paper we present stRDF triples that come from a fire monitoring and burnt area mapping application of project TELEIOS. The prefix `noa` used refers to the namespace of relevant vocabulary[5] defined by the National Observatory of Athens (NOA) for this application.

Example 1. stRDF triples derived from GeoNames that represent information about the Greek town Olympia including an approximation of its geometry. Also, stRDF triples that represent burnt areas.

```
geonames:26 rdf:type dbpedia:Town. geonames:26 geonames:name "Olympia".
geonames:26 strdf:hasGeometry "POLYGON((21 18,23 18,23 21,21 21,21 18));
                 <http://www.opengis.net/def/crs/EPSG/0/4326>"^^strdf:WKT.
noa:BA1 rdf:type noa:BurntArea;
        strdf:hasGeometry "POLYGON((0 0,0 2,2 2,2 0,0 0))"^^strdf:WKT.
noa:BA2 rdf:type noa:BurntArea;
        strdf:hasGeometry "POLYGON((3 8,4 9,3 9,3 8))"^^strdf:WKT.
```

Features can, in general, have many geometries (e.g., for representing the feature at different scales). Domain modelers are responsible for their appropriate utilization in their graphs and queries.

3 A New Version of stSPARQL

In this section we present a new version of the query language stSPARQL that we originally introduced in [9]. The new version is an extension of SPARQL 1.1 with functions that take as arguments spatial terms and can be used in the `SELECT`, `FILTER`, and `HAVING` clause of a SPARQL 1.1 query. A *spatial term* is either a spatial literal (i.e., a typed literal with datatype `strdf:geometry` or its subtypes), a query variable that can be bound to a spatial literal, the result of a set operation on spatial literals (e.g., union), or the result of a geometric operation on spatial terms (e.g., buffer).

In stSPARQL we use functions from the "OpenGIS Simple Feature Access - Part 2: SQL Option" standard (OGC-SFA)[6] for querying stRDF data. This standard defines relational schemata that support the storage, retrieval, query and update of sets of simple features using SQL.

A *feature* is a domain entity that can have various attributes that describe spatial and non-spatial (thematic) characteristics. The spatial characteristics of

[5] `http://www.earthobservatory.eu/ontologies/noaOntology.owl`
[6] `http://portal.opengeospatial.org/files/?artifact_id=25354`

a feature are represented using geometries such as points, lines, polygons, etc. Each geometry is associated with a CRS. A *simple feature* is a feature with all spatial attributes described piecewise by a straight line or a planar interpolation between sets of points. The OGC-SFA standard defines functions for requesting a specific representation of a geometry (e.g., the function `ST_AsText` returns the WKT representation of a geometry), functions for checking whether some condition holds for a geometry (e.g., the function `ST_IsEmpty` returns true if a geometry is empty) and functions for returning some properties of the geometry (e.g., the function `ST_Dimension` returns its inherent dimension). In addition, the standard defines functions for testing named spatial relationships between two geometries (e.g., the function `ST_Overlaps`) and functions for constructing new geometries from existing geometries (e.g., the function `ST_Envelope` that returns the minimum bounding box of a geometry).

The new version of stSPARQL extends SPARQL 1.1 with the machinery of the OGC-SFA standard. We achieve this by defining a URI for each of the SQL functions defined in the standard and use them in SPARQL queries. For example, for the function `ST_IsEmpty` defined in the OGC-SFA standard, we introduce the SPARQL extension function

`xsd:boolean strdf:isEmpty(strdf:geometry g)`

which takes as argument a spatial term `g`, and returns `true` if `g` is the empty geometry. Similarly, we have defined a Boolean SPARQL extension function for each topological relation defined in OGC-SFA (topological relations for simple features), [5] (Egenhofer relations) and [4] (RCC-8 relations). In this way stSPARQL supports multiple families of topological relations our users might be familiar with. Using these functions stSPARQL can express *spatial selections*, i.e., queries with a `FILTER` function with arguments a variable and a constant (e.g., `strdf:contains(?geo, "POINT(1 2)"^^strdf:WKT)`), and *spatial joins*, i.e., queries with a `FILTER` function with arguments two variables (e.g., `strdf:contains(?geoA, ?geoB)`).

The stSPARQL extension functions can also be used in the `SELECT` clause of a SPARQL query. As a result, new spatial literals can be generated on the fly during query time based on pre-existing spatial literals. For example, to obtain the buffer of a spatial literal that is bound to the variable `?geo`, we would use the expression `SELECT (strdf:buffer(?geo,0.01) AS ?geobuffer)`. In stSPARQL we have also the following three *spatial aggregate functions*:

– `strdf:geometry strdf:union(set of strdf:geometry a)`, returns a geometry that is the union of the set of input geometries.
– `strdf:geometry strdf:intersection(set of strdf:geometry a)`, returns a geometry that is the intersection of the set of input geometries.
– `strdf:geometry strdf:extent(set of strdf:geometry a)`, returns a geometry that is the minimum bounding box of the set of input geometries.

stSPARQL also supports update operations (insertion, deletion, and update of stRDF triples) conforming to the declarative update language for SPARQL, SPARQL Update 1.1, which is a current proposal of W3C.

The following examples demonstrate the functionality of stSPARQL.

Example 2. Return the names of towns that have been affected by fires.

```
SELECT  ?name
WHERE { ?t a dbpedia:Town; geonames:name ?name; strdf:hasGeometry ?tGeo.
        ?ba a noa:BurntArea; strdf:hasGeometry ?baGeo.
        FILTER(strdf:intersects(?tGeo,?baGeo))}
```

The query above demonstrates how to use a topological function in a query. The results of this query are the names of the towns whose geometries "spatially overlap" the geometries corresponding to areas that have been burnt.

Example 3. Isolate the parts of the burnt areas that lie in coniferous forests.

```
SELECT ?ba (strdf:intersection(?baGeom,strdf:union(?fGeom)) AS ?burnt)
WHERE { ?ba a noa:BurntArea. ?ba strdf:hasGeometry ?baGeom.
        ?f a noa:Area. ?f noa:hasLandCover noa:ConiferousForest.
        ?f strdf:hasGeometry ?fGeom.
        FILTER(strdf:intersects(?baGeom,?fGeom)) }
GROUP BY ?ba ?baGeom
```

The query above tests whether a burnt area intersects with a coniferous forest. If this is the case, groupings are made depending on the burnt area. The geometries of the forests corresponding to each burnt area are unioned, and their intersection with the burnt area is calculated and returned to the user. Note that only `strdf:union` is an aggregate function in the `SELECT` clause; `strdf:intersection` performs a computation involving the result of the aggregation and the value of `?baGeom` which is one of the variables determining the grouping according to which the aggregate computation is performed.

More details of stRDF and stSPARQL are given in [11].

4 Implementation

Strabon 3.0 is a fully-implemented, open-source, storage and query evaluation system for stRDF/stSPARQL and the corresponding subset of GeoSPARQL. We concentrate on stSPARQL only, but given the similarities with GeoSPARQL to be discussed in Section 6, the applicability to GeoSPARQL is immediate. Strabon has been implemented by extending the widely-known RDF store Sesame. We chose Sesame because of its open-source nature, layered architecture, wide range of functionalities and the ability to have PostGIS, a 'spatially enabled' DBMS, as a backend to exploit its variety of spatial functions and operators. Strabon is implemented by creating a layer that is included in Sesame's software stack in a transparent way so that it does not affect its range of functionalities, while benefitting from new versions of Sesame. Strabon 3.0 uses Sesame 2.6.3 and comprises three modules: the *storage manager*, the *query engine* and *PostGIS*.

The storage manager utilizes a bulk loader to store stRDF triples using the "one table per predicate" scheme of Sesame and dictionary encoding. For each predicate table, two B+ tree two-column indices are created. For each dictionary table a B+ tree index on the *id* column is created. All spatial literals are also stored in a table with schema *geo_values(id int, value geometry, srid int)*. Each tuple in the *geo_values* table has an *id* that is the unique encoding of the spatial literal based on the mapping dictionary. The attribute *value* is a spatial column whose data type is the PostGIS type `geometry` and is used to store the geometry that is described by the spatial literal. The geometry is transformed to a uniform, user-defined CRS and the original CRS is stored in the attribute *srid*. Additionally, a B+ tree index on the *id* column and an R-tree-over-GiST spatial index on the *value* column are created.

Query processing in Strabon is performed by the query engine which consists of a parser, an optimizer, an evaluator and a transaction manager. The parser and the transaction manager are identical to the ones in Sesame. The optimizer and the evaluator have been implemented by modifying the corresponding components of Sesame as we describe below.

The query engine works as follows. First, the parser generates an abstract syntax tree. Then, this tree is mapped to the internal algebra of Sesame, resulting in a query tree. The query tree is then processed by the optimizer that progressively modifies it, implementing the various optimization techniques of Strabon. Afterwards, the query tree is passed to the evaluator to produce the corresponding SQL query that will be evaluated by PostgreSQL. After the SQL query has been posed, the evaluator receives the results and performs any post-processing actions needed. The final step involves formatting the results. Besides the standard formats offered by RDF stores, Strabon offers KML and GeoJSON encodings, which are widely used in the mapping industry.

We now discuss how the optimizer works. First, it applies all the Sesame optimizations that deal with the standard SPARQL part of an stSPARQL query (e.g., it pushes down `FILTER`s to minimize intermediate results etc.). Then, two optimizations specific to stSPARQL are applied. The first optimization has to do with the extension functions of stSPARQL. By default, Sesame evaluates these *after* all bindings for the variables present in the query are retrieved. In Strabon, we modify the behaviour of the Sesame optimizer to incorporate all extension functions present in the `SELECT` and `FILTER` clause of an stSPARQL query into the query tree prior to its transformation to SQL. In this way, these extension functions will be evaluated using PostGIS spatial functions instead of relying on external libraries that would add an unneeded post-processing cost. The second optimization makes the underlying DBMS aware of the existence of spatial joins in stSPARQL queries so that they would be evaluated efficiently. Let us consider the query of Example 2. The first three triple patterns of the query are related to the rest via the topological function `strdf:intersects`. The query tree that is produced by the Sesame optimizer, fails to deal with this spatial join appropriately. It will generate a Cartesian product for the third and the fifth triple pattern of the query, and the evaluation of the spatial predicate `strdf:intersects` will be wrongly postponed after the calculation of this Cartesian product. Using a query graph as an intermediate representation of the query, we identify such spatial joins and modify the query tree with appropriate nodes so that Cartesian products are avoided. For the query of Example 2, the modified query tree will result in a SQL query that contains a θ-join where θ is the spatial function `ST_Intersects`.

More details of the query engine including examples of query trees and the SQL queries produced are given in the long version of this paper[7].

To quantify the gains of the optimization techniques used in Strabon, we also developed a *naive, baseline implementation* of stSPARQL which has none of the optimization enhancements to Sesame discussed above. The naive implementation uses the native store of Sesame as a backend instead of PostGIS (since the native store outperforms all other Sesame implementations using a DBMS). Data is stored on disk using atomic operations and indexed using B-Trees.

5 Experimental Evaluation

This section presents a detailed evaluation of the system Strabon using two different workloads: a workload based on linked data and a synthetic workload. For both workloads, we compare the response time of Strabon on top of PostgreSQL (called Strabon PG from now on) with our closest competitor implementation in [3], the naive, baseline implementation described in Section 4, and the RDF store Parliament. To identify potential benefits from using a different DBMS as a relational backend for Strabon, we also executed the SQL queries produced by Strabon in a proprietary spatially-enabled DBMS (which we will call System X, and Strabon X the resulting combination).

[3] presents an implementation which enhances the RDF-3X triple store with the ability to perform spatial selections using an R-tree index. The implementation of [3] is not a complete system like Strabon and does not support a full-fledged query language such as stSPARQL. In addition, the only way to load data in the system is the use of a generator which has been especially designed for the experiments of [3] thus it cannot be used to load other datasets in the implementation. Moreover, the geospatial indexing support of this implementation is limited to spatial selections. Spatial selections are pushed down in the query tree (i.e., they are evaluated before other operators). Parliament is an RDF storage engine recently enhanced with GeoSPARQL processing capabilities [2], which is coupled with Jena to provide a complete RDF system.

Our experiments were carried out on an Ubuntu 11.04 installation on an Intel Xeon E5620 with 12MB L2 cache running at 2.4 GHz. The system has 16GB of RAM and 2 disks of striped RAID (level 0). We measured the response time for each query posed by measuring the elapsed time from query submission till a complete iteration over the results had been completed. We ran all queries five times on cold and warm caches. For warm caches, we ran each query once before measuring the response time, in order to warm up the caches.

5.1 Evaluation using Linked Data

This section describes the experiments we did to evaluate Strabon using a workload based on linked data. We combined multiple popular datasets that include

[7] http://www.strabon.di.uoa.gr/files/Strabon-ISWC-2012-long-version.pdf

Dataset	Size	Triples	Spatial terms	Distinct spatial terms	Points	Linestrings (min/ max/ avg # of points/ linestring)	Polygons (min/max/avg # of points/ polygon)
DBpedia	7.1 GB	58,727,893	386,205	375,087	375,087	-	-
GeoNames	2.1GB	17,688,602	1,262,356	1,099,964	1,099,964	-	-
LGD	6.6GB	46,296,978	5,414,032	5,035,981	3,205,015	353,714 (4/20/9)	1,704,650 (4/20/9)
Pachube	828KB	6,333	101	70	70	-	-
SwissEx	33MB	277,919	687	623	623	-	-
CLC	14GB	19,711,926	2,190,214	2,190,214	-	-	2,190,214 (4/1,255,917/129)
GADM	146MB	255	51	51	-	-	51 (96/ 510,018/ 79,831)

Fig. 1. Summary of unified linked datasets

geospatial information: DBpedia, GeoNames, LinkedGeoData, Pachube, Swiss Experiment. We also used the datasets Corine Land Use/Land Cover and Global Administrative Areas that were published in RDF by us in the context of the EU projects TELEIOS and SemsorGrid4Env. The size of the unified dataset is 30GB and consists of 137 million triples that include 176 million distinct RDF terms, of which 9 million are spatial literals. As we can see in Table 1, the complexity of the spatial literals varies significantly. More information on the datasets mentioned and the process followed can be found in the long version of the paper. It should be noted that we could not use the implementation of [3] to execute this workload since it is not a complete system like Strabon as we explained above.

Storing stRDF Documents. For our experimental evaluation, we used the bulk loader mentioned in Section 4 emulating the "per-predicate" scheme since preliminary experiments indicated that the query response times in PostgreSQL were significantly faster when using this scheme compared to the "monolithic" scheme. This decision led to the creation of 48,405 predicate tables. Figure 2(a) presents the time required by each system to store the unified dataset.

The difference in times between Strabon and the naive implementation is natural given the very large number of predicate tables that had to be produced, processed and indexed. In this dataset, 80% of the total triples used only 17 distinct predicates. The overhead imposed to process the rest 48,388 predicates cancels the benefits of the bulk loader. A hybrid solution storing triples with popular predicates in separate tables while storing the rest triples in a single table, would have been more appropriate for balancing the tradeoff between storage time and query response time but we have not experimented with this option. In Section 5.2 we show that the loader scales much better when fewer distinct predicates are present in our dataset, regardless its size.

In the case of Strabon X, we followed the same process with Strabon PG. System X required double the amount of time that it took PostgreSQL to store and index the data. In the case of Parliament we modified the dataset to conform to GeoSPARQL and measured the time required for storing the resulting file. After incorporating the additional triples required by GeoSPARQL, the resulting dataset had approximately 15% more triples than the original RDF file.

Evaluating stSPARQL Queries. In this experiment we chose to evaluate Strabon using eight real-world queries. Our intention was to have enough queries to demonstrate Strabon's performance and functionality based on the following criteria: query patterns should be frequently used in Semantic Web applications or demonstrate the spatial extensions of Strabon. The first criterion was fulfilled by taking into account the results of [14] when designing the queries. [14] presents statistics on real-world SPARQL queries based on the logs of DBpedia. We also studied the query logs of the LinkedGeoData endpoint (which were kindly provided to us) to determine what kind of spatial queries are popular. According to this criterion, we composed the queries Q1, Q2 and Q7. According to [14] queries like Q1 and Q2 that consist of a few triple patterns are very common. Query Q7 consists of few triple patterns and a topological `FILTER` function, a structure similar to the queries most frequently posed to the LGD endpoint. The second criterion was fulfilled by incorporating spatial selections (Q4 and Q5) and spatial joins (Q3,Q6,Q7 and Q8) in the queries. In addition, we used non-topological functions that create new geometries from existing ones (Q4 and Q8) since these functions are frequently used in geospatial relational databases. In Table 2(b) we report the response time results. In queries Q1 and Q2, the baseline implementation outperforms all other systems as these queries do not include any spatial function. As mentioned earlier, the native store of Sesame outperforms Sesame implementations on top of a DBMS. All other systems produce comparable result times for these non-spatial queries. In all other queries the DBMS-based implementations outperform Parliament and the naive implementation. Strabon PG outperforms the naive implementation since it incorporates the two stSPARQL-specific optimizations discussed in Section 4. Thus, spatial operations are evaluated by PostGIS using a spatial index, instead of being evaluated after all the results have been retrieved. The naive implementation and Parliament fail to execute Q3 as this query involves a spatial join that is very expensive for systems using a naive approach. The only exception in the behavior of the naive implementation is Q4 in the case of warm caches, where the non-spatial part of the query produces very few results and the file blocks needed for query evaluation are cached in main memory. In this case, the non-spatial part of the query is executed rapidly while the evaluation of the spatial function over the results thus far is not significant. All queries except Q8 are executed significantly faster when run using Strabon on warm caches. Q8 involves many triple patterns and spatial functions which result in the production of a large number of intermediate results. As these do not fit in the system's cache, the response time is unaffected by the cache contents. System X decides to ignore the spatial index in queries Q3, Q6-Q8 and evaluate any spatial predicate exhaustively over the results of a thematic join. In queries Q6-Q8, it also uses Cartesian products in the query plan as it considers their evaluation more profitable. These decisions are correct in the case of Q8 where it outperforms all other systems significantly, but very costly in the cases of Q3, Q6 and Q7.

System	Total time (sec)			
	Linked Data	**10mil**	**100mil**	**500mil**
Naive	9,480	1,053	12,305	72,433
Strabon PG	19,543	458	3,241	21,155
Strabon X	28,146	818	7,274	40,378
RDF-3X	*	780	8,040	43,201
Parliament	81,378	734	6,415	>36h

(a)

Caches	**System**	**Q1**	**Q2**	**Q3**	**Q4**	**Q5**	**Q6**	**Q7**	**Q8**
Cold (sec.)	Naive	0.08	1.65	>8h	28.88	89	170	844	1.699
	Strabon-PG	2.01	6.79	41.39	10.11	78.69	60.25	9.23	702.55
	Strabon-X	1.74	3.05	1623.57	46.52	12.57	2409.98	>8h	57.83
	Parliament	2.12	6.46	>8h	229.72	1130.98	872.48	3627.62	3786.36
Warm (sec.)	Naive	0.01	0.03	>8h	0.79	43.07	88	708	1712
	Strabon-PG	0.01	0.81	0.96	1.66	38.74	1.22	2.92	648.1
	Strabon-X	0.01	0.26	1604.9	35.59	0.18	3196.78	>8h	44.72
	Parliament	0.01	0.04	>8h	10.91	358.92	483.29	2771	3502.53

(b)

Fig. 2. (a) Storage time for each dataset (b) Response time for real-world queries

5.2 Evaluation Using a Synthetic Dataset

Although we had tested Strabon with datasets up to 137 million triples (Section 5.1), we wanted better control over the size and the characteristics of the spatial dataset being used for evaluation. By using a generator to produce a synthetic dataset, we could alter the thematic and spatial selectivities of our queries and closely monitor the performance of our system, based on both spatial and thematic criteria. Since we could produce a dataset of arbitrary size, we were able to stress our system by producing datasets of size up to 500 million triples.

Storing stRDF Documents. The generator we used to produce our datasets is a modified version of the generator used by the authors of [3], which was kindly provided to us. The data produced follows a general version of the schema of Open Street Map depicted in Figure 3. Each node has a spatial extent (the location of the node) and is placed uniformly on a grid. By modifying the step of the grid, we produce datasets of arbitrary size. In addition, each node is assigned a number of tags each of which consists of a key-value pair of strings. Every node is tagged with key 1, every second node with key 2, every fourth node with key 4, etc. up to key 1024. We generated three datasets consisting of 10 million, 100 million and 500 million triples and stored them in Strabon using the per-predicate scheme. Figure 2(a) shows the time required to store these datasets. In the case of [3], the generator produces directly a binary file using the internal format of RDF-3X and computes exhaustively all indices, resulting in higher storage times. On the contrary, Strabon, Parliament and the naive implementation store an RDF file for each dataset. We observed that Strabon's bulk loader is very efficient when dealing with any dataset not including an excessive amount of scarcely used distinct predicates, and regardless of the underlying DBMS. In the case of Parliament, the dataset resulting from the conversion to GeoSPARQL was 27% bigger than the original 100 million triples dataset. Its storage time was shorter than that of all other systems but Strabon PG. However, Parliament failed to store the 500 million triples dataset after 36 hours.

Evaluating stSPARQL Queries. In this experiment we used the following query template that is identical to the query template used in [3]:

```
SELECT * WHERE {?node geordf:hasTag ?tag. ?node strdf:hasGeography ?geo.
?tag geordf:key PARAM_A. FILTER (strdf:inside(?geo, PARAM_B))}
```

In this query template, `PARAM_A` is one of the values used when tagging a node and `PARAM_B` is the WKT representation of a polygon. We define the

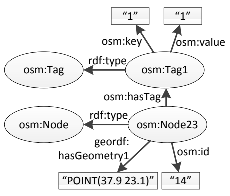


Fig. 3. LGD schema

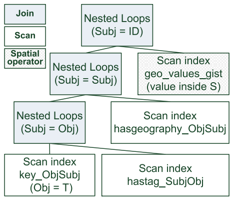

Join
Scan
Spatial operator
Nested Loops (Subj = ID)
Nested Loops (Subj = Subj)
Scan index geo_values_gist (value inside S)
Nested Loops (Subj = Obj)
Scan index hasgeography_ObjSubj
Scan index key_ObjSubj (Obj = T)
Scan index hastag_SubjObj

Fig. 4. Plan A

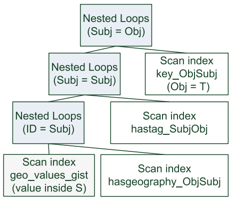
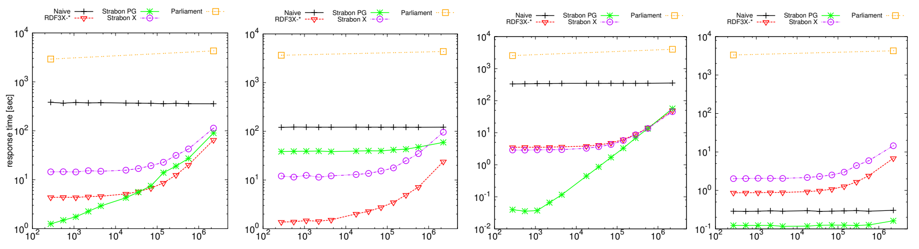

Nested Loops (Subj = Obj)
Nested Loops (Subj = Subj)
Scan index key_ObjSubj (Obj = T)
Nested Loops (ID = Subj)
Scan index hastag_SubjObj
Scan index geo_values_gist (value inside S)
Scan index hasgeography_ObjSubj

Fig. 5. Plan B

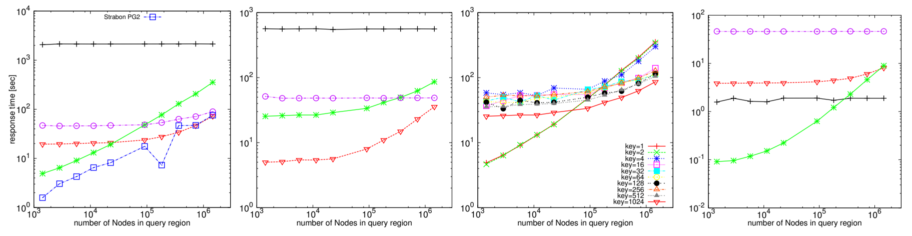
thematic selectivity of an instantiation of the query template as the fraction of the total nodes that are tagged with a key equal to PARAM_A. For example, by altering the value of PARAM_A from 2 to 4, we reduce the thematic selectivity of the query by selecting half the nodes we previously did. We define the *spatial selectivity* of an instantiation of the query template as the fraction of the total nodes that are inside the polygon defined by PARAM_B. We modify the size of the polygon in order to select from 10 up to 10^6 nodes.

We will now discuss representative experiments with the 100 and 500 million triples datasets. For each dataset we present graphs depicting results based on various combinations of parameters PARAM_A and PARAM_B in the case of cold or warm caches. These graphs are presented in Figure 6. In the case of Strabon, the stSPARQL query template is mapped to a SQL query, which is subsequently executed by PostgreSQL or System X. This query involves the predicate tables `key(subj,obj)`, `hasTag(subj,obj)` and `hasGeography(subj,obj)` and the spatial table `geo_values(id,value,srid)`. We created two B+ tree two-column indices for each predicate table and an R-tree index on the `value` column of the spatial table. The main point of interest in this SQL query is the order of join execution in the following sub-query: $\sigma_{obj=T}(key) \bowtie hastag \bowtie \sigma_{value\ inside\ S}(hasgeography)$ where T and S are values of the parameters PARAM_A and PARAM_B respectively. Different orders give significant differences in query execution time. After consulting the query logs of PostgreSQL, we noticed that the vast majority of the SQL queries that were posed and derived from the query template adhered to one of the query plans of Figures 4,5. According to plan A, query evaluation starts by evaluating the thematic selection over table `key` using the appropriate index. The results are retrieved and joined with the `hasTag` and `hasGeography` predicate tables using appropriate indices. Finally, the spatial selection is evaluated by scanning the spatial index and the results are joined with the intermediate results of the previous operations. On the contrary, plan B starts with the evaluation of the spatial selection, leaving the application of the thematic selection for the end.

Unfortunately, in the current version of PostGIS spatial selectivities are not computed properly. The functions that estimate the selectivity of a spatial selection/join return a constant number regardless of the actual selectivity of the operator. Thus, only the thematic selectivity affects the choice of a query plan. To state it in terms of our query template, altering the value PARAM_A between 1 and 1024 was the only factor influencing the selection of a query plan.



(a) 100 mil.,tag 1 (cold) (b) 100 mil.,tag 1024 (cold) (c) 100 mil.,tag 1 (warm) (d) 100 mil.,tag 1024(warm)



(e) 500 mil.,tag 1 (cold) (f) 500 mil.,tag 1024(cold) (g) 500 mil. (cold) (h) 500 mil.,tag 1024(warm)

Fig. 6. Response times

A representative example is shown in Figure 6(g), where we present the response times for all values of PARAM_A and observe two patterns. When thematic selectivity is high (values 1-2), the response time is low when the spatial selectivity (captured by the x-axis) is low, while it increases along with the spatial selectivity. This happens because for these values PostgreSQL chooses plan B, which is good only when the spatial selectivity is low. In cases of lower thematic selectivity (values 4-1024), the response time is initially high but only increases slightly as spatial selectivity increases. In this case, PostgreSQL chooses plan A, which is good only when the spatial selectivity is high. The absence of dynamic estimation of spatial selectivity is affecting the system performance since it does not necessarily begin with a good plan and fails to switch between plans when the spatial selectivity passes the turning point observed in the graph.

Similar findings were observed for System X. The decision of which query plan to select was influenced by PARAM_A. In most cases, a plan similar to plan B is selected, with the only variation that in the case of the 500 million triples dataset the access methods defined in plan B are full table scans instead of index scans. The only case in which System X selected another plan was when posing a query with very low thematic selectivity (PARAM_A = 1024) on the 500 million triples dataset, in which case System X switched to a plan similar to plan A.

In Figures 6(a)-6(d) we present the results for the 100 million dataset for the extreme values 1 and 1024 of `PARAM_A`. Strabon PG outperforms the other systems in the case of warm caches, while the implementation of [3] and Strabon X outperform Strabon PG for value 1024 in the case of cold caches (Figure 6(b)). In this case, as previously discussed, PostgreSQL does not select a good plan and the response times are even higher than the times where `PARAM_A` is equal to 1 (Figure 6(a)), despite producing significantly more intermediate results.

In Figures 6(e), 6(f) and 6(h) we present the respective graphs for the 500 million dataset and observe similar behavior as before, with a small deviation in Figure 6(e), where the implementation of [3] outperforms Strabon. In this scenario, Strabon X also gradually outperforms Strabon PG as spatial selectivity increases, although PostgreSQL had chosen the correct query plan. Taking this into account, we tuned PostgreSQL to make better use of the system resources. We allowed the usage of more shared buffers and the allocation of larger amounts of memory for internal sort operations and hash tables. As observed in Figure 6(e), the result of these modifications (labeled Strabon PG2) led to significant performance improvement. This is quite typical with relational DBMS; the more you know about their internals and the more able you are to tune them for queries of interest, the better performance you will get.

In general, Strabon X performs well in cases of high spatial selectivity. In Figure 6(f), the response time of Strabon X is almost constant. In this case a plan similar to plan A is selected, with the variation that the spatial index is not utilized for the evaluation of the spatial predicate. This decision would be correct only for queries with high spatial selectivity. The effect of this plan is more visible in Figure 6(h), where the large number of intermediate results prevent the system to utilize its caches, resulting in constant, yet high response times.

Regarding Parliament, although the results returned by each query posed were correct, the fact that the response time was very high did not allow us to execute queries using all instantiations of the query template. We instantiated the query template using the extreme values of `PARAM_A` and `PARAM_B` and executed them to have an indication of the system's performance.

In all datasets used, we observe that the naive implementation has constant performance regardless of the spatial selectivity of a query since the spatial operator is evaluated against all bindings retrieved thus far. The baseline implementation also outperforms System X and the implementation of [3] in queries with low thematic selectivity (value 1024) over warm caches.

In summary, Strabon PG outperforms the other implementations when caches are warmed up. Results over cold caches are mixed, but we showed that aggressive tuning of PostgreSQL can increase the performance of Strabon resulting to response times slightly better than the implementation of [3]. Nevertheless, modifications to the optimizer of PostgreSQL are needed in order to estimate accurately the spatial selectivity of a query to produce better plans by combining this estimation with the thematic selectivity of the query. Given the significantly better performance of RDF-3X over Sesame for standard SPARQL queries, another open question in our work is how to go beyond [3] and modify the optimizer of RDF-3X so that it can deal with the full stSPARQL query language.

6 Related Work

Geospatial extensions of RDF and SPARQL have been presented recently in [7,9,13]. An important addition to this line of research is the recent OGC candidate standard GeoSPARQL discussed in [1] and the new version of stRDF/stSPARQL presented in this paper. Like stRDF/stSPARQL, GeoSPARQL aims at providing a basic framework for the representation and querying of geospatial data on the Semantic Web. The two approaches have been developed independently at around the same time, and have concluded with very similar representational and querying constructs. Both approaches represent geometries as literals of an appropriate datatype. These literals may be encoded in various formats like GML, WKT etc. Both approaches map spatial predicates and functions that support spatial analysis to SPARQL extension functions. GeoSPARQL goes beyond stSPARQL in that it allows binary topological relations to be used as RDF properties anticipating their possible utilization by spatial reasoners (this is the topological extension and the related query rewrite extension of GeoSPARQL). In our group, such geospatial reasoning functionality is being studied in the more general context of "incomplete information in RDF" of which a preliminary overview is given in [10]. Since stSPARQL has been defined as an extension of SPARQL 1.1, it goes beyond GeoSPARQL by offering geospatial aggregate functions and update statements that have not been considered at all by GeoSPARQL. Another difference between the two frameworks is that GeoSPARQL imposes an RDFS ontology for the representation of features and geometries. On the contrary, stRDF only asks that a specific literal datatype is used and leaves the responsibility of developing any ontology to the users. In the future, we expect user communities to develop more specialized ontologies that extend the basic ontologies of GeoSPARQL with relevant geospatial concepts from their own application domain. In summary, strictly speaking, stSPARQL and GeoSPARQL are incomparable in terms of representational power. If we omit aggregate functions and updates from stSPARQL, its features are a subset of the features offered by the GeoSPARQL core, geometry extension and geometry topology extension components. Thus, it was easy to offer full support in Strabon for these three parts of GeoSPARQL.

Recent work has also considered implementation issues for geospatial extensions of RDF and SPARQL. [3] presents an implementation based on RDF-3X, which we already discussed in detail. Virtuoso provides support for the representation and querying of two-dimensional point geometries expressed in multiple reference systems. Virtuoso models geometries by typed literals like stSPARQL and GeoSPARQL. Vocabulary is offered for a subset of the SQL/MM standard to perform geospatial queries using SPARQL. The open-source edition does not incorporate these geospatial extensions. [2] introduces GeoSPARQL and the implementation of a part of it in the RDF store Parliament. [2] discusses interesting ideas regarding query processing in Parliament but does not give implementation details or a performance evaluation. A more detailed discussion of current proposals for adding geospatial features to RDF stores is given in our recent survey [8]. None of these proposals goes beyond the expressive power of stSPARQL or GeoSPARQL discussed in detail above.

7 Conclusions and Future Work

We presented the new version of the data model stRDF and the query language stSPARQL, the system Strabon which implements the complete functionality of stSPARQL (and, therefore, a big subset of GeoSPARQL) and an experimental evaluation of Strabon. Future work concentrates on experiments with even larger datasets, comparison of Strabon with other systems such as Virtuoso, and stSPARQL query processing in the column store MonetDB.

References

1. Open Geospatial Consortium. OGC GeoSPARQL - A geographic query language for RDF data. OGC Candidate Implementation Standard (Februaey 2012)
2. Battle, R., Kolas, D.: Enabling the Geospatial Semantic Web with Parliament and GeoSPARQL. In: Semantic Web Interoperability, Usability, Applicability (2011)
3. Brodt, A., Nicklas, D., Mitschang, B.: Deep integration of spatial query processing into native RDF triple stores. In: ACM SIGSPATIAL (2010)
4. Cohn, A., Bennett, B., Gooday, J., Gotts, N.: Qualitative Spatial Representation and Reasoning with the Region Connection Calculus. Geoinformatica (1997)
5. Egenhofer, M.J.: A Formal Definition of Binary Topological Relationships. In: Litwin, W., Schek, H.-J. (eds.) FODO 1989. LNCS, vol. 367, pp. 457–472. Springer, Heidelberg (1989)
6. Gutierrez, C., Hurtado, C., Vaisman, A.: Introducing Time into RDF. IEEE Trans. Knowl. Data Eng. 19(2), 207–218 (2007)
7. Kolas, D., Self, T.: Spatially-Augmented Knowledgebase. 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.) ISWC/ASWC 2007. LNCS, vol. 4825, pp. 792–801. Springer, Heidelberg (2007)
8. Koubarakis, M., Karpathiotakis, M., Kyzirakos, K., Nikolaou, C., Sioutis, M.: Data Models and Query Languages for Linked Geospatial Data. In: Eiter, T., Krennwallner, T. (eds.) Reasoning Web 2012. LNCS, vol. 7487, pp. 290–328. Springer, Heidelberg (2012)
9. Koubarakis, M., Kyzirakos, K.: Modeling and Querying Metadata in the Semantic Sensor Web: The Model stRDF and the Query Language stSPARQL. 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. 425–439. Springer, Heidelberg (2010)
10. Koubarakis, M., Kyzirakos, K., Karpathiotakis, M., Nikolaou, C., Sioutis, M., Vassos, S., Michail, D., Herekakis, T., Kontoes, C., Papoutsis, I.: Challenges for Qualitative Spatial Reasoning in Linked Geospatial Data. In: BASR (2011)

11. Koubarakis, M., Kyzirakos, K., Nikolaou, B., Sioutis, M., Vassos, S.: A data model and query language for an extension of RDF with time and space. Del. 2.1, TELEIOS
12. Kuper, G., Ramaswamy, S., Shim, K., Su, J.: A Constraint-based Spatial Extension to SQL. In: ACM-GIS, pp. 112–117 (1998)
13. Perry, M.: A Framework to Support Spatial, Temporal and Thematic Analytics over Semantic Web Data. Ph.D. thesis, Wright State University (2008)
14. Picalausa, F., Vansummeren, S.: What are real SPARQL queries like? In: International Workshop on Semantic Web Information Management (2011)

DeFacto - Deep Fact Validation*

Jens Lehmann, Daniel Gerber, Mohamed Morsey,
and Axel-Cyrille Ngonga Ngomo

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

Abstract. One of the main tasks when creating and maintaining knowledge bases is to validate facts and provide sources for them in order to ensure correctness and traceability of the provided knowledge. So far, this task is often addressed by human curators in a three-step process: issuing appropriate keyword queries for the statement to check using standard search engines, retrieving potentially relevant documents and screening those documents for relevant content. The drawbacks of this process are manifold. Most importantly, it is very time-consuming as the experts have to carry out several search processes and must often read several documents. In this article, we present DeFacto (Deep Fact Validation) – an algorithm for validating facts by finding trustworthy sources for it on the Web. DeFacto aims to provide an effective way of validating facts by supplying the user with relevant excerpts of webpages as well as useful additional information including a score for the confidence DeFacto has in the correctness of the input fact.

1 Introduction

The past decades have been marked by a change from an industrial society to an information and knowledge society. This change is particularly due to the uptake of the World Wide Web. Creating and managing knowledge successfully has been a key to success in various communities worldwide. Therefore, the quality of knowledge is of high importance. One aspect of knowledge quality is provenance. In particular, the sources for facts should be well documented, since this provides several benefits such as a better detection of errors, decisions based on the trustworthiness of sources etc. While provenance is an important aspect of data quality [8], to date only few knowledge bases actually provide provenance information. For instance, less than 3% of the more than 607.7 million RDF documents indexed by Sindice[1] contain metadata such such as creator, created, source, modified, contributor, or provenance.[2] This lack of provenance

* This work was partially supported by a grant from the European Union's 7th Framework Programme provided for the project LOD2 (GA no. 257943) and Eurostars E!4604 SCMS.

[1] http://www.sindice.com

[2] Data retrieved on June 6, 2012.

information makes the validation of the facts in such knowledge bases utterly tedious. In addition, it hinders the adoption of such data in business applications as the data is not trusted [8]. The main contribution of this paper is the provision of a fact validation approach and tool which can make use of one of the largest sources of information: the Web.

More specifically, our system DeFacto (Deep Fact Validation) implements algorithms for validating RDF triples by finding confirming sources for it on the web. It takes a statement as input (e.g., that shown in Listing 1) and then tries to find evidence for the validity of that statement by searching for textual information in the web. In contrast to typical search engines, it does not just search for textual occurrences of parts of the statement, but tries to find webpages which contain the actual statement phrased in natural language. It presents the user with a confidence score for the input statement as well as a set of excerpts of relevant webpages, which allows the user to manually judge the presented evidence.

DeFacto has two major use cases: (1) Given an existing true statement, it can be used to find provenance information for it. For instance, the WikiData project[3] aims to create a collection of facts, in which sources should be provided for each fact. DeFacto could be used to achieve this task. (2) It can check whether a statement is likely to be true, provide the user with a confidence score in whether the statement is true and evidence for the score assigned to the statement. Our main contributions are thus as follows: (1) An approach that allows checking whether a webpage confirms a fact, i.e., an RDF triple, (2) an adaptation of existing approaches for determining indicators for trustworthiness of a webpage, (3) an automated approach to enhancing knowledge bases with RDF provenance data at triple level as well as (4) a running prototype of DeFacto, the first system able to provide useful confidence values for an input RDF triple given the Web as background text corpus.

The rest of this paper is structured as follows: Section 2 describes our general approach and the system infrastructure. The next section describes how we extended the BOA framework to enable it to detect facts contained in textual descriptions on webpages. In Section 4, we describe how we include the trustworthiness of webpages into the DeFacto analysis. Section 5 combines the results from the previous chapters and describes the mathematical features we use to compute the confidence for a particular input fact. We use those features to train different classifiers in Section 6 and describe our evaluation results. Section 7 summarizes related work. Finally, we conclude in Section 8 and give pointers to future work.

2 Approach

Input and Output: The DeFacto system consists of the components depicted in Figure 1. The system takes an RDF triple as input and returns a confidence value for this triple as well as possible evidence for the fact. The evidence consists of a

[3] `http://meta.wikimedia.org/wiki/Wikidata`

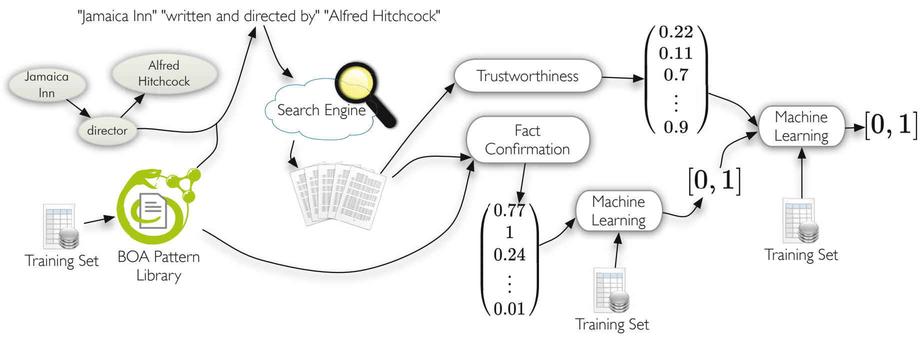


Fig. 1. Overview of Deep Fact Validation

set of webpages, textual excerpts from those pages and meta-information on the pages. The text excerpts and the associated meta information allow the user to quickly get an overview over possible credible sources for the input statement: Instead of having to use search engines, browsing several webpages and looking for relevant pieces of information, the user can more efficiently review the presented information. Moreover, the system uses techniques which are adapted specifically for fact validation instead of only having to rely on generic information retrieval techniques of search engines.

Retrieving Webpages: The first task of the DeFacto system is to retrieve webpages which are relevant for the given task. The retrieval is carried out by issuing several queries to a regular search engine. These queries are computed by verbalizing the RDF triple using natural-language patterns extracted by the BOA framework[4] [5,4]. Section 3.2 describes how the search engine queries are constructed. As a next step, the highest ranked webpages for each query are retrieved. Those webpages are candidates for being sources for the input fact. Both the search engine queries as well as the retrieval of webpages are executed in parallel to keep the response time for users within a reasonable limit. Note that usually this does not put a high load on particular web servers as webpages are usually derived from several domains.

Evaluating Webpages: Once all webpages have been retrieved, they undergo several further processing steps. First, plain text is extracted from each webpage by removing most HTML markup. We can then apply our fact confirmation approach on this text, which is described in detail in Section 3. In essence, the algorithm decides whether the web page contains a natural language formulation of the input fact. This step distinguishes DeFacto from information retrieval methods. If no webpage confirms a fact according to DeFacto, then the system falls back on lightweight NLP techniques and computes whether the webpage does at

[4] `http://boa.aksw.org`

least provide useful evidence. In addition to fact confirmation, the system computes different indicators for the trustworthiness of a webpage (see Section 4). These indicators are of central importance because a single trustworthy webpage confirming a fact may be a more useful source than several webpages with low trustworthiness. The fact confirmation and the trustworthiness indicators of the most relevant webpages are presented to the user.

Confidence Measurement: In addition to finding and displaying useful sources, DeFacto also outputs a general confidence value for the input fact. This confidence value ranges between 0% and 100% and serves as an indicator for the user: Higher values indicate that the found sources appear to confirm the fact and can be trusted. Low values mean that not much evidence for the fact could be found on the Web and that the websites that do confirm the fact (if such exist) only display low trustworthiness. The confidence measurement is based on machine learning techniques and explained in detail in Sections 5 and 6. Naturally, DeFacto is a (semi-)automatic approach: We do assume that users will not blindly trust the system, but additionally analyze the provided evidence.

Using the LOD Cloud as Background Knowledge: As described above, DeFacto relies primarily on natural language from several webpages as input. However, in some cases, confirming facts for an input statement can be found in openly available knowledge bases. Due to the fast growth of the LOD cloud, we expect this source to become increasingly important in the future. In order to use this additional evidence, DeFacto provides a preliminary component which searches for similar statements in the LOD cloud. To achieve this goal, the system first finds similar resources to the subject and object of the input triple, which is currently done via the `http://sameas.org` service. In a second step, it retrieves all triples which use the detected similar subject and object resources by dereferencing the corresponding Linked Data URIs. Finally, the labels of subject, predicate and object of all triples are retrieved. Those are then compared via string similarity techniques to the input triple. Currently, the average trigram similarity of subject, predicate and object of the triple is used. In this article, we focus on re-using textual evidence and plan to carry out a more detailed evaluation of the LOD as background knowledge in future work.

```
dbpedia-res:Jamaica_Inn_%28film%29 dbpedia-owl:director
      dbpedia-res:Alfred_Hitchcock .
```

Listing 1. Input data for Defacto

A prototype implementing the above steps is available at `http://defacto.aksw.org`. It shows relevant webpages, text excerpts and five different rankings per page. The generated provenance output can also be saved directly as RDF. For representing the provenance output, we use the W3C provenance group[5] vocabularies. The source code of both, the DeFacto algorithms and user interface, are openly available[6].

[5] `http://www.w3.org/2011/prov/`

[6] `https://github.com/AKSW/DeFacto`

It should be noted that we decided not to check for negative evidence of facts in DeFacto, since a) we considered this to be too error-prone and b) negative statements are much less frequent on the web. It is also noteworthy that DeFacto is a self training system on two levels: For each fact, the user can confirm after reviewing the possible sources whether he believes it is true. This is then added to the training set and helps to improve the performance of DeFacto. The same can be done for text excerpts of web pages: Users can confirm or reject whether a given text excerpt actually does confirm a fact. Both machine learning parts are explained in Sections 3 and 6.

3 BOA

The idea behind BOA is two-fold: first, it aims to be a framework that allows extracting structured data from the Human Web by using Linked Data as background knowledge. In addition, it provides a library of natural-language patterns that allows to bridge the gap between structured and unstructured data. The input for the BOA framework consists of a set of knowledge bases, a text corpus (mostly extracted from the Web) and (optionally) a Wikipedia dump[7]. When provided by a Wikipedia dump, the framework begins by generating surface forms for all entities in the source knowledge base. The surface forms used by BOA are generated by using an extension of the method proposed in [12]. For each predicate p found in the input knowledge sources, BOA carries out a sentence-level statistical analysis of the co-occurrence of pairs of labels of resources that are linked via p. BOA then uses a supervised machine-learning approach to compute a score and rank patterns for each combination of corpus and knowledge bases. These patterns allow generating a natural-language representation of the RDF triple that is to be checked.

3.1 Training BOA for DeFacto

In order to provide a high quality fact confirmation component, we trained BOA specifically for this task. We began by selecting the top 60 most frequently used object properties from the DBpedia [13,10] ontology using the DBpedia Live endpoint[8]. This query retrieves 7,750,362 triples and covers 78% of the 9,993,333 triples in DBpedia with *owl:ObjectProperty*s from the DBpedia namespace.[9] Currently, we focus on object properties. Adequate support of datatype properties requires an extension of the presented methods, which is planned in future work. For each of those properties, we selected the top 10 BOA patterns (if available) sorted according to the number of triples this pattern has been learned from. This resulted in a list of 488 patterns which were evaluated by all four authors. During this process, each pattern was labeled by two persons

[7] http://dumps.wikimedia.org/

[8] http://live.dbpedia.org/sparql

[9] Properties like wikiPageExternalLink, wikiPageRedirects, wikiPageDisambiguates and thumbnail have been excluded.

independently. We judged a pattern as positive if it was not generic (e.g., "?D? 's " ?R?") or specific enough (e.g., "?D? in the Italian region ?R?") and could be used to express the relation in natural text. The first group achieved a moderate Cohen's-Kappa value of 0.477 and the second group scored a good value of 0.626. Every conflict was resolved by having the annotators agree on a single annotation. The resulting annotations were used for a 10-fold cross-validation training of BOA's neural network. We achieved the maximum F-score of 0.732 with an error threshold of 0.01 and a hidden layer size of 51 neurons.

3.2 Automatic Generation of Search Queries

The found BOA patterns are used for issuing queries to the search engine (see Figure 1). Each search query contains the quoted label of the subject of the input triple, a quoted and cleaned BOA pattern (we remove punctuation characters which are not indexed by the search engine) and the quoted label of the object of the input triple. We use a fixed number of the best-scored BOA patterns whose score was beyond a score threshold and retrieve the first n websites from a web search engine. For our example from Listing 1, an examplary query sent to the search engine is `‘‘Jamaican Inn’’ AND ‘‘written and directed by’’ AND ‘‘Alfred Hitchcock’’`. We then crawl each website and try to extract *possible proofs* for the input triple, i.e., excerpts of these webpages which may confirm it. For the sake of brevity, we use proof and possible proof interchangeably.

3.3 BOA and NLP Techniques for Fact Confirmation

To find proofs for a given input triple we make use of the surface forms introduced in [12]. We select all surface forms for the subject and object of the input triple and search for all occurrences of each combination of those labels in a website w. If we find an occurrence with a token distance $d(l(s), l(o))$ (where $l(x)$ is the label of x) smaller then a given threshold we call this occurrence a proof for the input triple. To remove noise from the found proofs we apply a set of normalizations by using regular expression filters which for example remove characters between brackets and non alpha-numeric characters. Note that this normalization improves the grouping of proofs by their occurrence. After extracting all proofs $p_i \in \mathcal{P}(w)$ of a website w, we score each proof using a linear regression classifier. We trained a classifier with the following input features for scoring a proof:

BOA Pattern: This is a Boolean feature which is 1 if a BOA pattern is contained in the normalized proof phrase.

BOA Score: If BOA patterns are found in the normalized proof phrase, then the score of the highest score across the set of found patterns is written in this feature. Else, this feature is set to 0.

Token Distance: This is the distance $d(l(s), l(o))$ between the two entity labels which found the proof. We limit this distance to a maximum of 20 tokens.

Table 1. Performance measures for several classifiers on the fact confirmation task (AUC = area under the ROC curve, RMSE = root mean squared error)

	P	R	F_1	AUC	RMSE
Logistic Regression	0.769	0.769	0.769	0.811	0.4653
Naïve Bayes	0.655	0.624	0.564	0.763	0.5665
SVM	0.824	0.822	**0.822**	0.823	0.4223
RBFNetwork	0.735	0.717	0.718	0.718	0.485

Wordnet Expansion: We expand both the tokens of the normalized proof phrase as well as all of the tokens of the BOA pattern with synsets from Wordnet. Subsequently we apply the Jaccard-Similarity on the generated expansions. This is basically a fuzzy match between the BOA pattern and the proof phrase.

Total Occurrence: This feature contains the total number of occurrences of each normalized proof phrase over the set of all normalized proof phrases.

Page Title: We apply the maximum of the trigram similarity measure between the page title and the subject and object labels. This feature is useful, because the title indicates the topic of the entire web page. When a title matches, then higher token distances may still indicate a high probability that a fact is confirmed.

End of Sentence: A boolean value if the potential proof contains a ".", "!" or a "?". When subject and object are in different sentences, their relation is more likely to be weaker.

Phrase: The words between the subject and object, which are encoded as binary values, i.e. a feature is created for each word and its value is set to 1 if the word occurs and 0 otherwise.

Property: The property as a word vector.

To train our classifiers, we randomly sampled 527 proofs and annotated them manually. Those proofs were extracted with DeFacto from applying it on the training set described in Section 6.1. The results are shown in Table 1. We ran popular classifiers, which are able to work with numeric data and create confidence values. The ability to generate confidence values for proofs is useful as feedback for users and it also serves as input for the core classifiers described in Section 6. Based on the obtained results, we selected support vector machines as classifier. We also performed preliminary work on fine-tuning the parameters of the above algorithms, which, however, did not lead to significantly different results. The reported measurements were, therefore, done with default values of the mentioned algorithms in the Weka machine learning toolkit[10] version 3.6.6.

4 Trustworthiness Analysis of Webpages

To determine the trustworthiness of a website we first need to determine its similarity to the input triple. This is determined by how many topics belonging

[10] http://www.cs.waikato.ac.nz/ml/weka/

to the query are contained in a search result retrieved by the web search. We extended the approach introduced in [14] by querying Wikipedia with the subject and object label of the triple in question separately to find the topic terms for the triple. A frequency analysis is applied on all returned documents and all terms above a certain threshold that are not contained in a self-compiled stop word list are considered to be topic terms for a triple. Let s and o be the URIs for the subject and object of the triple in question and t be a potential topic term extracted from a Wikipedia page. In addition, let $X = (s, p, o)$.We compare the values of the following two formulas:

$$p(t|X) = \frac{|topic(t, d(X))|}{|d(X)|},$$
$$p(t|intitle(d(X), s \vee o)) = \frac{|topic(t, intitle(d(X), s) \cup intitle(d(X), o))|}{|intitle(d(X), s) \cup intitle(d(X), o)|}.$$

where $d(X)$ is the set all web documents retrieved for X (see Section 3.2), $intitle(d(X), x)$ the set of web documents which have the label of the URI x in their page title. $topic(t, d(X))$ is the set of documents which contain t in the page body. We consider t to be a topic term for the input triple if $p(t|t(d(X), s) \vee t(d(X), o)) > p(t|X)$. Let $\mathcal{T}_X = \{t_1, t_2, \ldots, t_n\}$ be the set of all topic terms extracted for a input triple. Defacto then calculates the trustworthiness of a webpage as follows:

Topic Majority in the Web represents the number of webpages that have similar topics to the webpage in question. Let $\mathcal{P}$ be the set of topic terms appearing on the current webpage. The Topic Majority in the Web for a webpage w is then calculated as:

$$tm_{web}(w) = \left| \bigcup_{i=1}^{n} topic(t_i, d(X)) \right| - 1.$$

where t_1 is the most occurring topic term in the webpage w. Note that we subtract 1 to prevent counting w.

Topic Majority in Search Results calculates the similarity of a given webpage for all webpages found for a given triple. Let w_k be the webpage to be evaluated, $v(w_k)$ be the feature vector of webpage w_k where $v(w_k)_i$ is 1 if t_i is a topic term of webpage w_k and 0 otherwise, $\|v\|$ be the norm of v and θ a similarity threshold. We calculate the Topic Majority for the search results as follows:

$$tm_{search}(w) = \left| \left\{ w_i | w_i \in d(X), \frac{v(w_k) \times v(w_i)}{\|v(w_k)\| \, \|v(w_i)\|} > \theta \right\} \right|.$$

Topic Coverage measures the ratio between all topic terms for X and all topic terms occurring in w:

$$tc(w) = \frac{|\mathcal{T}_X \cap \mathcal{P}|}{|\mathcal{T}_X|}.$$

Pagerank: The Pagerank[11] of a webpage is a measure for the relative importance of a webpage compared to all others, i.e. higher pageranks means that a webpage is more popular. There is a positive correlation between popularity of a webpage and its trustworthiness as those pages are more likely to be reviewed by more people or may have gone under stricter quality assurance before their publication.While a high pagerank alone is certainly not a sufficient indicator for trustworthiness, we use it in combination with the above criteria in DeFacto.

5 Features for Deep Fact Validation

In order to obtain an estimate of the confidence that there is sufficient evidence to consider the input triple to be true, we decided to train a supervised machine learning algorithm. Similar to the above presented classifier for fact confirmation, this classifier also requires computing a set of relevant features for the given task. In the following, we describe those features and why we selected them.

First, we extend the score of single proofs to a score of web pages as follows: When interpreting the score of a proof as the probability that a proof actually confirms the input fact, then we can compute the probability that at least one of the proofs confirms the fact. This leads to the following stochastic formula[12], which allows us to obtain an overall score for proofs scw on a webpage w:

$$scw(w) = 1 - \prod_{pr \in prw(w)} (1 - fc(pr)) \, .$$

In this formula, fc (fact confirmation) is the classifier trained in Section 3.3, which takes a proof pr as input and returns a value between 0 and 1. prw is a function taking a webpage as input and returning all possible proofs contained in it.

Combination of Trustworthiness and Textual Evidence In general, the trustworthiness of a webpage and the textual evidence we find in it, are orthogonal features. Naturally, webpages with high trustworthiness and a high score for its proofs should increase our confidence in the input fact. Therefore, it makes sense to combine trustworthiness and textual evidence as features for the underlying machine learning algorithm. We do this by multiplying both criteria and then using their sum and maximum as two different features:

$$F_{fsum}(t) = \sum_{w \in s(t)} (f(w) \cdot scw(w)) \qquad F_{fmax}(t) = \max_{w \in s(t)} (f(w) \cdot scw(w))$$

In this formula f can be instantiated by all four trustworthiness measures: topic majority on the the web (tm_{web}), topic majority in search results (tm_{search}),

[11] http://en.wikipedia.org/wiki/Pagerank

[12] To be exact, it is the complementary even to the case that none of the proofs do actually confirm a fact.

topic coverage (tc) and pagerank (pr). s is a function taking a triple t as argument, executing the search queries explained in Section 3.2 and returning a set of webpages. Using the formula, we obtain 8 different features for our classifier, which combine textual evidence and different trustworthiness measures.

Other Features. In addition to the above described combinations of trustworthiness and fact confirmation, we also defined other features:

1. The total number of proofs found.
2. The total number of proofs found above a relevance threshold of 0.5. In some cases, a high number of proofs with low scores is generated, so the number of high scoring proofs may be a relevant feature for learning algorithms.
3. The total evidence score: This is the probability that at least one of the proofs is correct, which is defined analogously to *scw* above:
$$1 - \prod_{pr \in prt(t)} (1 - sc(pr)) \, .$$
4. The total evidence score above a relevance threshold of 0.5. This is an adaption of the above formula, which considers only proofs with a confidence higher than 0.5.
5. Total hit count: Search engines usually estimate the number of search results for an input query. The total hit count is the sum of the estimated number of search results for each query send by DeFacto for a given input triple.
6. A domain and range verification: If the subject of the input triple is not an instance of the domain of the property of the input triple, this violates the underlying schema, which should result in a lower confidence in the correctness of the triple. This feature is 0 if both domain and range are violated, 0.5 if exactly one of them is violated and 1 if there is no domain or range violation.

6 Evaluation

Our main objective in the evaluation was to find out whether DeFacto can effectively distinguish between true and false input facts. In the following, we describe how we trained DeFacto using DBpedia, which experiments we used and discuss the results of our experiments.

6.1 Training DeFacto

As mentioned in Section 3, we focus our experiments on the top 60 most frequently used properties in DBpedia. The system can easily be extended to cover more properties by extending the training set of BOA to those properties. Note that DeFacto itself is also not limited to DBpedia, i.e., while all of its components are trained on DBpedia, the algorithms can be applied to arbitrary URIs. A performance evaluation on other knowledge bases is subject to future work, but it should be noted that most parts of DeFacto – except the LOD background feature described in Section 2 and the schema checking feature in Section 5 work only with the retrieved labels of URIs and, therefore, do not depend on DBpedia.

For training a supervised machine learning approach, positive and negative examples are required. Those were generated as follows:

Positive Examples: In general, we use facts contained in DBpedia as positive examples. For each of the properties we consider (see Section 3), we generated positive examples by randomly selecting triples containing the property. We collected 600 statements in this manner and verified them by checking manually whether it was indeed a true fact. It turned out that some of the obtained triples were modeled incorrectly, e.g. obviously violated domain and range restrictions or could not be confirmed by an intensive search on the web within ten minutes. Overall, 473 out of 600 checked triples were facts could be used as positive examples.

Negative Examples: The generation of negative examples is more involved than the generation of positive examples. In order to effectively train DeFacto, we considered it essential that many of the negative examples are similar to true statements. In particular, most statements should be meaningful triples. For this reason, we derived the negative examples from positive examples by modifying them while still following domain and range restrictions. Assume the input triple (s, p, o) in a knowledge base $\mathcal{K}$ is given and let *dom* and *ran* be functions returning the domain and range of a property[13]. We used the following methods to generate the negative example sets dubbed *subject*, *object*, *subject-object*, *property*, *random*, *20%mix* (in that order):

1. A triple (s', p, o) is generated where s' is an instance of $dom(p)$, the triple (s', p, o) is not contained in $\mathcal{K}$ and s' is randomly selected from all resources which satisfy the previous requirements.
2. A triple (s, p, o') is generated analogously by taking $ran(p)$ into account.
3. A triple (s', p, o') is generated analogously by taking both $dom(p)$ and $ran(p)$ into account.
4. A triple (s, p', o) is generated in which p' is randomly selected from our previously defined list of 60 properties and (s, p', o) is not contained in $\mathcal{K}$.
5. A triple (s', p', o') is generated where s' and o' are randomly selected resources, p' is a randomly selected property from our defined list of 60 properties and (s', p', o') is not contained in $\mathcal{K}$.
6. 20% of each of the above created negative training sets were randomly selected to create a heterogenous test set.

Note that all parts of the example generation procedure can also take implicit knowledge into account, e.g., by simply extending our approach to use SPARQL 1.1 entailment[14]. In case of DBpedia Live we did not use any entailment this for

[13] Technically, we used the most specific class, which was explicitly stated to be domain and range of a property, respectively.

[14] http://www.w3.org/TR/sparql11-entailment/

performance reasons and because it would not alter the results in that specific case.

Obviously, it is possible that our procedure for generating negative examples also generates true statements which just happen not to be contained in DBpedia. Similar to the analysis of the positive examples, we checked a sample of the negative examples on whether they are indeed false statements. This was the case for all examples in the sample. Overall, we obtained an automatically created and manually cleaned training set, which we made publicly available[15].

6.2 Experimental Setup

In a first step, we computed all feature vectors, described in Section 5 for the training set. DeFacto relies heavily on web requests, which are not deterministic (i.e. the same search engine query does not always return the same result). To achieve deterministic behavior and to increase the performance as well as reduce load on the servers, all web requests are cached. The DeFacto runtime for an input triple was on average slightly below 5 seconds per input triple[16] when using caches.

We stored the features in the arff file format and employed the Weka machine learning toolkit[17] for training different classifiers. In particular, we were interested in classifiers which can handle numeric values and output confidence values. Naturally, confidence values for facts such as, e.g. 95%, are more useful for end users than just a binary response on whether DeFacto considers the input triple to be true, since they allow a more fine-grained assessment. Again, we selected popular machine-learning algorithms satisfying those requirements.

We performed 10-fold cross-validations for our experiments. In each experiment, we used our created positive examples, but varied the negative example sets described above to see how changes influence the overall behavior of DeFacto.

Table 2. Classification results for trainings sets *subject* and *object*

	Subject					Object				
	P	R	F_1	AUC	RSME	P	R	F_1	AUC	RMSE
Logistic Regression	0.799	0.753	0.743	0.83	0.4151	0.881	0.86	0.859	0.844	0.3454
Naïve Bayes	0.739	0.606	0.542	0.64	0.6255	0.795	0.662	0.619	0.741	0.5815
SVM	0.811	0.788	0.784	0.788	0.4609	0.884	0.867	**0.865**	0.866	0.3409
J48	0.835	0.827	**0.826**	0.819	0.3719	0.869	0.862	0.861	0.908	0.3194
RBF Network	0.743	0.631	0.583	0.652	0.469	0.784	0.683	0.652	0.75	0.4421

[15] http://aksw.org/projects/DeFacto

[16] The performance is roughly equal on server machines and notebooks, since the web requests dominate.

[17] http://www.cs.waikato.ac.nz/ml/weka/

Table 3. Classification results for trainings sets *subject-object* and *property*

	Subject-Object					Property				
	P	R	F_1	AUC	RSME	P	R	F_1	AUC	RMSE
Logistic Regression	0.871	0.85	0.848	0.86	0.3495	0.822	0.818	0.818	0.838	0.3792
Naïve Bayes	0.813	0.735	0.717	0.785	0.5151	0.697	0.582	0.511	0.76	0.6431
SVM	0.88	0.863	0.861	0.855	0.3434	0.819	0.816	0.816	0.825	0.3813
J48	0.884	0.871	**0.87**	0.901	0.3197	0.834	0.832	**0.832**	0.828	0.3753
RBF Network	0.745	0.687	0.667	0.728	0.4401	0.72	0.697	0.688	0.731	0.4545

Table 4. Classification results for trainings sets *random* and *20%mix*

	Random					20% Mix				
	P	R	F_1	AUC	RMSE	P	R	F_1	AUC	RMSE
Logistic Regression	0.855	0.854	0.854	0.908	0.3417	0.665	0.645	0.634	0.785	0.4516
Naïve Bayes	0.735	0.606	0.544	0.853	0.5565	0.719	0.6	0.538	0.658	0.6267
SVM	0.855	0.854	0.854	0.906	0.3462	0.734	0.729	0.728	0.768	0.4524
J48	0.876	0.876	**0.876**	0.904	0.3226	0.8	0.79	**0.788**	0.782	0.405
RBF Network	0.746	0.743	0.742	0.819	0.4156	0.698	0.61	0.561	0.652	0.4788

6.3 Results and Discussion

The results of our experiments are shown in Tables 2-4. Three algorithms – J48, logistic regression and support vector machines – show promising results. Given the challenging tasks, F-measures up to 78.8% for the combined negative example set appear to be very positive indicators that DeFacto can be used to effectively distinguish between true and false statements, which was our primary evaluation objective. In general, DeFacto also appears to be stable against the various negative example sets. In particular, the algorithms with overall positive results also seem less affected by the different variations. When observing single runs of DeFacto manually, it turned out that our method of generating positive examples is particularly challenging for DeFacto: For many of the facts in DBpedia only few sources exist in the Web. While it is widely acknowledged that the amount of unstructured textual information in the Web by far surpasses the available structured data, we found out that a significant amount of statements in DBpedia is difficult to track back to reliable external sources on the Web even with an exhaustive manual search. There are many reasons for this, for instance many facts are particular relevant for a specific country, such as "Person x studied at University y.", where x is a son of a local politician and y is a country with only limited internet access compared to first world countries. For this reason, BOA patterns could be only be detected directly in 29 of the 527 proofs of positive examples. This number increased to 195 out of 527 when we employed the WordNet expansion described in Section 3.3. In general, DeFacto performs better when the subject and object of the input triple are popular on the web, i.e. there are several webpages describing them. In this aspect, we believe our training set is indeed challenging upon manual observation.

7 Related Work

While we are not aware of existing work in which sources for RDF statements were detected automatically from the Web, there are three main areas related to DeFacto research: The representation of provenance information in the Web of Data as well as work on trustworthiness and relation extraction. The problem of data provenance is a crucial issue in the Web of Data. While data extracted by the means of tools such as Hazy[18] and KnowItAll[19] can be easily mapped to primary provenance information, most knowledge sources were extracted by non-textual source and are more difficult to link with provenance information. In the work described in [9], Olaf Hartig and Jun Zhao developed a framework for provenance tracking. This framework provides the vocabulary required for representing and accessing provenance information on the web. It keeps track of who created a web entity, e.g. a webpage, when it was last modified etc. Recently, a W3C working group has been formed and released a set of specifications on sharing and representing provenance information[20]. Dividino et al. [3] introduced an approach for managing several provenance dimensions, e.g. source, and timestamp. In their approach, they describe an extension to the RDF called RDF$^+$ which can work efficiently with provenance data. They provided a method to extend SPARQL query processing in a manner such that a specific SPARQL query can request meta knowledge without modifying the query itself. Theoharis et al. [18] argued how the implicit provenance data contained in a SPARQL query results can be used to acquire annotations for several dimensions of data quality. They detailed the abstract provenance models and how they are used in relational data, and how they can be used in semantic data as well. Their model requires the existence of provenance data in the underlying semantic data source. DeFacto uses the W3C provenance group standard for representing provenance information. Yet, unlike previous work, it directly tries to find provenance information by searching for confirming facts in trustworthy webpages.

The second related research area is trustworthiness. Nakamura et al. [14] developed an efficient prototype for enhancing the search results provided by a search engine based on trustworthiness analysis for those results. They conducted a survey in order to determine the frequency at which the users accesses search engines and how much they trust the content and ranking of search results. They defined several criteria for trustworthiness calculation of search results returned by the search engine, such as topic majority. We adapted their approach for DeFacto and included it as one of the features for our machine learning techniques. [16,17] present an approach for computing the trustworthiness of web pages. To achieve this goal, the authors rely on a model based on hubs and authorities.

[18] http://hazy.cs.wisc.edu/hazy/

[19] http://www.cs.washington.edu/research/knowitall/

[20] http://www.w3.org/2011/prov/wiki/

This model allows to compute the trustworthiness of facts and websites by generating a k-partite network of pages and facts and propagating trustworthiness information across it. The approach returns a score for the trustworthiness of each fact. An older yet similar approach is that presented in [20]. Here, the idea is to create a 3-partite network of webpages, facts and objects and apply a propagation algorithm to compute weights for facts as well as webpages. The use of trustworthiness and uncertainty information on RDF data has been the subject of recent research (see e.g., [7,11]). Our approach differs from these approaches as it does not aim to evaluate the trustworthiness of facts expressed in natural language. In addition, it can deal with the broad spectrum of relations found on the Data Web.

Most tools that address this task rely on pattern-based approaches. Some early work on pattern extraction relied on supervised machine learning [6]. Our approach is also related to relation extraction. Yet, such approaches demanded large amounts of training data, making them difficult to adapt to new relations. The subsequent generation of approaches to RE aimed at bootstrapping patterns based on a small number of input patterns and instances. For example, [2] presents the Dual Iterative Pattern Relation Expansion (DIPRE) and applies it to the detection of relations between authors and titles of books. This approach relies on a small set of seed patterns to maximize the precision of the patterns for a given relation while minimizing their error rate of the same patterns. Snowball [1] extends DIPRE by a new approach to the generation of seed tuples. Newer approaches aim to either collect redundancy information (see e.g., [19]) in an unsupervised manner or to use linguistic analysis [15] to harvest generic patterns for relations.

8 Conclusion and Future Work

In this paper, we presented DeFacto, an approach for checking the validity of RDF triples using the Web as corpus. We showed that our approach achieves an average F_1 measure (J48 for all 6 datasets) of 0.843 on DBpedia. Our approach can be extended in manifold ways. First, BOA is able to detect natural-language representations of predicates in several languages. Thus, we could have the user choose the languages he understands and provide facts in several languages, therewith also increasing the portion of the Web that we search through. Furthermore, we could extend our approach to support data type properties. In addition to extending our approach by these two means, we will also focus on searching for negative evidence for facts, therewith allowing users to have an unbiased view of the data on the Web through DeFacto. On a grander scale, we aim to provide even lay users of knowledge bases with the means to check the quality of their data by using natural language input.

References

1. Agichtein, E., Gravano, L.: Snowball: Extracting relations from large plain-text collections. In: ACM DL, pp. 85–94 (2000)

2. Brin, S.: Extracting Patterns and Relations from the World Wide Web. In: Atzeni, P., Mendelzon, A.O., Mecca, G. (eds.) WebDB 1998. LNCS, vol. 1590, pp. 172–183. Springer, Heidelberg (1999)
3. Dividino, R., Sizov, S., Staab, S., Schueler, B.: Querying for provenance, trust, uncertainty and other meta knowledge in rdf. Web Semantics: Science, Services and Agents on the World Wide Web 7(3) (2011)
4. Gerber, D., Ngomo, A.-C.N.: Extracting Multilingual Natural-Language Patterns for RDF Predicates. 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. 87–96. Springer, Heidelberg (2012)
5. Gerber, D., Ngomo, A.-C.N.: Bootstrapping the linked data web. In: 1st Workshop on Web Scale Knowledge Extraction ISWC (2011)
6. Grishman, R., Yangarber, R.: Nyu: Description of the Proteus/Pet system as used for MUC-7 ST. In: MUC-7. Morgan Kaufmann (1998)
7. Hartig, O.: Trustworthiness of data on the web. In: Proceedings of the STI Berlin & CSW PhD Workshop (2008)
8. Hartig, O.: Provenance information in the web of data. In: Proceedings of LDOW (2009)
9. Hartig, O., Zhao, J.: Publishing and Consuming Provenance Metadata on the Web of Linked Data. In: McGuinness, D.L., Michaelis, J.R., Moreau, L. (eds.) IPAW 2010. LNCS, vol. 6378, pp. 78–90. Springer, Heidelberg (2010)
10. Lehmann, J., Bizer, C., 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)
11. Meiser, T., Dylla, M., Theobald, M.: Interactive reasoning in uncertain RDF knowledge bases. In: Berendt, B., de Vries, A., Fan, W., Macdonald, C. (eds.) CIKM 2011, pp. 2557–2560 (2011)
12. 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)
13. 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)
14. Nakamura, S., Konishi, S., Jatowt, A., Ohshima, H., Kondo, H., Tezuka, T., Oyama, S., Tanaka, K.: Trustworthiness Analysis of Web Search Results. In: Kovács, L., Fuhr, N., Meghini, C. (eds.) ECDL 2007. LNCS, vol. 4675, pp. 38–49. Springer, Heidelberg (2007)
15. Nguyen, D.P.T., Matsuo, Y., Ishizuka, M.: Relation extraction from wikipedia using subtree mining. In: AAAI, pp. 1414–1420 (2007)
16. Pasternack, J., Roth, D.: Generalized fact-finding. In: WWW 2011, pp. 99–100 (2011)
17. Pasternack, J., Roth, D.: Making better informed trust decisions with generalized fact-finding. In: IJCAI, pp. 2324–2329 (2011)
18. Theoharis, Y., Fundulaki, I., Karvounarakis, G., Christophides, V.: On provenance of queries on semantic web data. IEEE Internet Computing 15, 31–39 (2011)
19. Yan, Y., Okazaki, N., Matsuo, Y., Yang, Z., Ishizuka, M.: Unsupervised relation extraction by mining wikipedia texts using information from the web. In: ACL 2009, pp. 1021–1029 (2009)
20. Yin, X., Han, J., Yu, P.S.: Truth discovery with multiple conflicting information providers on the web. In: KDD 2007, pp. 1048–1052 (2007)

Feature LDA: A Supervised Topic Model for Automatic Detection of Web API Documentations from the Web

Chenghua Lin, Yulan He, Carlos Pedrinaci, and John Domingue

Knowledge Media Institute, The Open University
Milton Keynes, MK7 6AA, UK
{c.lin,y.he,c.pedrinaci,j.b.domingue}@open.ac.uk

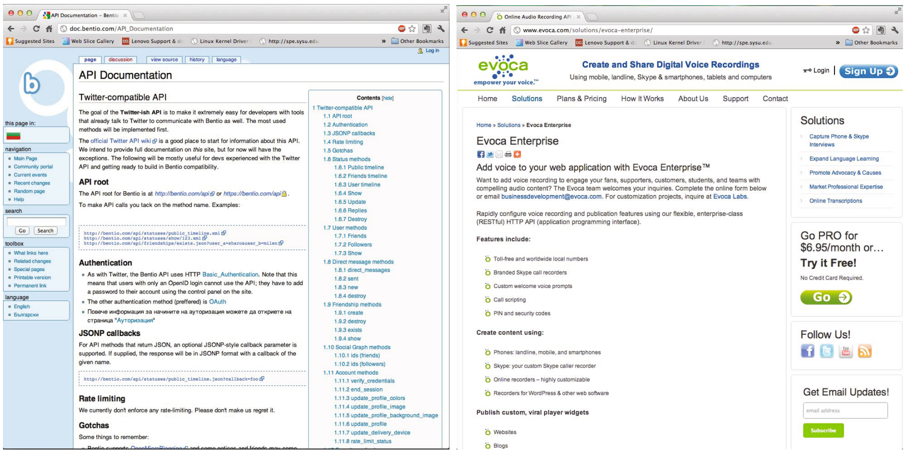
Abstract. Web APIs have gained increasing popularity in recent Web service technology development owing to its simplicity of technology stack and the proliferation of mashups. However, efficiently discovering Web APIs and the relevant documentations on the Web is still a challenging task even with the best resources available on the Web. In this paper we cast the problem of detecting the Web API documentations as a text classification problem of classifying a given Web page as Web API associated or not. We propose a supervised generative topic model called feature latent Dirichlet allocation (feaLDA) which offers a generic probabilistic framework for automatic detection of Web APIs. feaLDA not only captures the correspondence between data and the associated class labels, but also provides a mechanism for incorporating side information such as labelled features automatically learned from data that can effectively help improving classification performance. Extensive experiments on our Web APIs documentation dataset shows that the feaLDA model outperforms three strong supervised baselines including naive Bayes, support vector machines, and the maximum entropy model, by over 3% in classification accuracy. In addition, feaLDA also gives superior performance when compared against other existing supervised topic models.

1 Introduction

On the Web, service technologies are currently marked by the proliferation of Web APIs, also called RESTful services when they conform to REST principles. Major Web sites such as Facebook, Flickr, Salesforce or Amazon provide access to their data and functionality through Web APIs. To a large extent this trend is impelled by the simplicity of the technology stack, compared to WSDL and SOAP based Web services, as well as by the simplicity with which such APIs can be offered over preexisting Web site infrastructures [15].

When building a new service-oriented application, a fundamental step is discovering existing services or APIs. Main means used nowadays by developers for locating Web APIs are searching through dedicated registries like ProgrammableWeb[1]

[1] http://www.programmableweb.com/



API Documentation

Twitter-compatible API

The goal of the **Twitter-ish API** is to make it extremely easy for developers with tools that already talk to Twitter to communicate with Bentio as well. The most used methods will be implemented first.

The official Twitter API wiki is a good place to start for information about this API. We intend to provide full documentation on *this* site, but for now will have the exceptions. The following will be mostly useful for devs experienced with the Twitter API and getting ready to build in Bentio compatibility.

API root

The API root for Bentio is at *http://bentio.com/api* or *https://bentio.com/api*.

To make API calls you tack on the method name. Examples:

```
http://bentio.com/api/statuses/public_timeline.xml
http://bentio.com/api/statuses/show/123.xml
http://bentio.com/api/friendships/exists.json?user_a=sharo&user_b=milen
```

Authentication

- As with Twitter, the Bentio API uses HTTP Basic_Authentication. Note that this means that users with only an OpenID login cannot use the API; they have to add a password to their account using the control panel on the site.
- The other authentication method (preffered) is OAuth
- Повече информация за начините на ауторизация можете да откриете на страница "Ауторизация"

JSONP callbacks

For API methods that return JSON, an optional JSONP-style callback parameter is supported. If supplied, the response will be in JSONP format with a callback of the given name.

```
http://bentio.com/api/statuses/public_timeline.json?callback=foo
```

Rate limiting

We currently don't enforce any rate-limiting. Please don't make us regret it.

Gotchas

Some things to remember:

Contents [hide]

(a) True API documentation.

Create and Share Digital Voice Recordings

Using mobile, landline, Skype & smartphones, tablets and computers

Evoca Enterprise

Add voice to your web application with Evoca Enterprise™

Want to add voice recording to engage your fans, supporters, customers, students, and teams with compelling audio content? The Evoca team welcomes your inquiries. Complete the online form below or email businessdevelopment@evoca.com. For customization projects, inquire at Evoca Labs.

Rapidly configure voice recording and publication features using our flexible, enterprise-class (RESTful) HTTP API (application programming interface).

Features include:

- Toll-free and worldwide local numbers
- Branded Skype call recorders
- Custom welcome voice prompts
- Call scripting
- PIN and security codes

Create content using:

- Phones: landline, mobile, and smartphones
- Skype: your custom Skype caller recorder
- Online recorders – highly customizable
- Recorders for WordPress & other web software

Publish custom, viral player widgets

- Websites
- Blogs

Solutions

- Capture Phone & Skype Interviews
- Expand Language Learning
- Promote Advocacy & Causes
- Market Professional Expertise
- Online Transcriptions

Go PRO for $6.95/month or…

Try it Free!

No Credit Card Required.

(b) False API documentation.

Fig. 1. Examples of Web pages documenting and not documenting Web APIs

which are manually populated or to use traditional search engines like Google. While public API registries provide highly valuable information, there are also some noticeable issues. First, more often than not, these registries contain out of date information (e.g. closed APIs are still listed) or even provide incorrect links to APIs documentation pages (e.g. the home page of the company is given instead). Indeed, the manual nature of the data acquisition in APIs registries aggravates these problems as new APIs appear, disappear or change. Automatically detecting the incorrect information will help registry operator better maintain their registry quality and provide better services to developers. Second, partly due to the manual submission mechanism, APIs listed in the public registries are still limited where a large number of valuable third party Web APIs may not be included. In this case, the alternative approach is to resort to Web search engine. However, general purpose search engines are not optimised for this type of activity and often mix relevant pages documenting Web APIs with general pages e.g., blogs and advertisement. Figure 1 shows both a Web pages documenting an API and one that is not that relevant but would still be presented in the results returned by search engines.

Motivated by the above observations, in our ongoing work on iServe (a public platform for service publication and discovery), we are building an automatic Web APIs search engine for detecting third party Web APIs on the Web scale. Particularly, we assume that Web pages documenting APIs are good identifiers for the detection as whenever we use an API the first referring point is likely to be the related documentation. While identifying WSDL services are relatively easy by detecting the WSDL documentation which has a standard format, detecting Web APIs documentation raises more challenges. This is due to the fact that Web APIs are generally described in plain and unstructured HTML which are only readable by human being; and to make it worse, the format of documenting a Web API is highly heterogeneous, so as its content and level of details [11]. Therefore, a prerequisite to the success of our Web APIs search engine is to construct a classifier which can offer high performance in identifying Web pages documenting Web APIs.

In this paper, we propose a novel supervised topic model called feature latent Dirichlet allocation (feaLDA) for text classification by formulating the generative process that topics are draw dependent on document class labels and words are draw conditioned on the document label-topic pairs. Particularly, feaLDA is distinguished from other related supervised topic models in its capability of accommodating different types of supervision. In particular, while supervised topic models such as labeled LDA and partial labeled LDA (pLDA) [19,20] can only model the correspondence between class labels and documents, feaLDA is able to incorporate supervision from both document labels and labelled features for effectively improving classification performance, where the labelled features can be learned automatically from training data.

We tested feaLDA on a Web APIs dataset consisting of 622 Web pages documenting Web APIs and 925 normal Web pages crawled from ProgrammingWeb. Results from extensive experiments show that the proposed feaLDA model can achieve a very high precision of 85.2%, and it significantly outperforms three strong supervised baselines (i.e. naive Bayes, maximum entropy and SVM) as well as two closed related supervised topic models (i.e. labeled LDA and pLDA) by over 3% in accuracy. Aside from text classification, feaLDA can also extract meaningful topics with clear class label associations as illustrated by some topic examples extracted from the Web APIs dataset.

The rest of the paper is organised as follows. Section 2 reviews the previous work on Web APIs detection and the supervised topic models that are closely related to feaLDA. Section 3 presents the feaLDA model and the model inference procedure. Experimental setup and results on the Web APIs dataset are discussed in Sections 4 and 5, respectively. Finally, Section 6 concludes the paper and outlines the future work.

2 Related Work

Web Service Discovery. Service discovery has been the subject of much research and development. The most renown work is perhaps Universal Description Discovery and Integration (UDDI) [3], while nowadays Seekda[2] provides the largest public index with about 29,000 WSDL Web services. The adoption of these registries has, however, been limited [3,18]. Centred around WSDL, UDDI and related service technologies, research on semantic Web services has generated a number of ontologies, semantic discovery engines, and further supporting infrastructure aiming at improving the level of automation and accuracy that can be obtained throughout the life-cycle of service-oriented application, see [17] for an extensive survey. Despite these advances, the majority of these initiatives

[2] `http://webservices.seekda.com/`

are predicated upon the use of WSDL Web services, which have turned out not to be prevalent on the Web where Web APIs are increasingly favoured [15].

A fundamental characteristic of Web APIs is the fact that, despite a number of proposals [6,7], there is no widely adopted means for publishing these services nor for describing their functionality in a way such that machines could automatically locate these APIs and understand the functionality and data they offer. Instead, Web APIs are solely accompanied by highly heterogeneous HTML pages providing documentation for developers. As a consequence, there has not been much progress on supporting the automated discovery of Web APIs. Perhaps the most popular directory of Web APIs is ProgrammableWeb which, as of June 2012, lists about 6,200 APIs and provides rather simple search mechanisms based on keywords, tags, or a simple prefixed categorisation. Based on the data provided by ProgrammableWeb, APIHut [5] increases the accuracy of keyword-based search of APIs compared to ProgrammableWeb or plain Google search. A fundamental drawback of ProgrammableWeb and by extension of APIHut is that they rely on the manual registration of APIs by users. This data tends to be out of date (e.g., discontinued APIs are still listed) and often provide pointers to generic Web pages (e.g., the home page of the company offering the API) which are not particularly useful for supporting the discovery and use of the related APIs. Finally, iServe [15] enables the application of advanced (semantic) discovery algorithms for Web API discovery but, thus far, it is limited by the fact that it relies on the presence of hRESTS annotations in Web pages which are still seldom available.

Therefore, despite the increasing relevance of Web APIs, there is hardly any system available nowadays that is able to adequately support their discovery. The first and main obstacle in this regard concerns the automated location of Web APIs, which is the main focus of this paper. In this regard, to the best of our knowledge, we are only aware of two previous initiatives. One was carried out by Steinmetz et al. [22], whose initial experiments are, according to the authors, not sufficiently performant and require further refinement. The second approach [16] is our initial work in this area which we herein expand and enhance.

Topic Models. As shown in previous work [4,12,21,25], topic models constructed for purpose-specific applications often involve incorporating side information or supervised information as prior knowledge for model learning, which in general can be categorised into two types depending on how the side information are incorporated [13]. One type is the so called *downstream* topic models, where both words and document metadata such as author, publication date, publication venue, etc. are generated simultaneously conditioned on the topic assignment of the document. Examples of this type include the mixed membership model [4] and the Group Topic (GT) model [25]. The *upstream* topic models, by contrast, start the generative process with the observed side information, and represent the topic distributions as a mixture of distributions conditioned on the side information elements. Examples of this type are the Author-Topic (AT) model [21] and the joint sentiment-topic (JST) model [9,10]. Although JST can detect sentiment and topic simultaneously from text by incorporating prior information to modify the Dirichlet priors of the topic-word distribution, it is still a weakly-supervised model as no mechanism is provided to incorporate document class label for model inference.

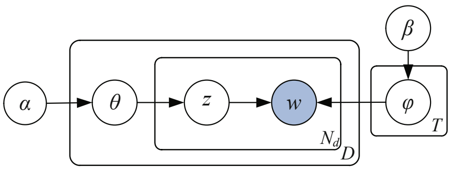
For both downstream and upstream models, most of the models are customised for a special type of side information, lacking the capability to accommodate data type beyond their original intention. This limitation has thus motivated work on developing a generalised framework for incorporating side information into topic models [2, 13]. The supervised latent Dirichlet allocation (sLDA) model [2] addresses the prediction problem of review ratings by inferring the most predictive latent topics of document labels. Mimno and McCallum [13] proposed the Dirichlet-multinomial regression (DMR) topic model which includes a log-linear prior on the document-topic distributions, where the prior is a function of the observed document features. The intrinsic difference between DMR and its complement model sLDA lies in that, while sLDA treats observed features as generated variables, DMR considers the observed features as a set of conditioned variables. Therefore, while incorporating complex features may result in increasingly intractable inference in sLDA, the inference in DMR can remain relatively simple by accounting for all the observed side information in the document-specific Dirichlet parameters.

Closely related to our work are the supervised topic models incorporating document class labels. DiscLDA [8] and labeled LDA [19] apply a transformation matrix on document class labels to modify Dirichlet priors of the LDA-like models. While labeled LDA simply defines a one-to-one correspondence between LDA's latent topics and observed document labels and hence does not support latent topics within a give document label, Partially Labeled LDA (pLDA) extends labeled LDA to incorporate per-label latent topics [20]. Different from the previous work where only document labels are incorporated as prior knowledge into model learning, we propose a novel feature LDA (feaLDA) model which is capable of incorporating supervised information derive from both the document labels and the labelled features learned from data to constrain the model learning process.

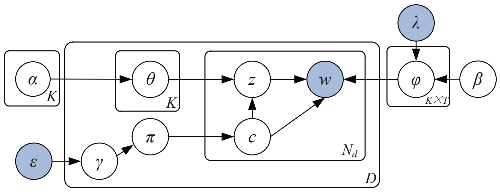
3 The Feature LDA (feaLDA) Model

The feaLDA model is a supervised generative topic model for text classification by extending latent Dirichlet allocation (LDA) [1] as shown in Figure 2a. feaLDA accounts for document labels during the generative process, where each document can associate with a single class label or multiple class labels. In contrast to most of the existing supervised topic models [8,19,20], feaLDA not only accounts for the correspondence between class labels and data, but can also incorporate side information from labelled features to constrain the Dirichlet prior of topic-word distributions for effectively improving classification performance. Here the labelled features can be learned automatically from training data using any feature selection method such as information gain.

The graphical model of feaLDA is shown in Figure 2b. Assume that we have a corpus with a document collection $D = \{d_1, d_2, ..., d_D\}$; each document in the



(a)



(b)

Fig. 2. (a) LDA model; (b) feaLDA model.

corpus is a sequence of N_d words denoted by $d = (w_1, w_2, ..., w_{N_d})$, and each word in the document is an item from a vocabulary index with V distinct terms. Also, letting K be the number of class labels, and T be the total number of topics, the complete procedure for generating a word w_i in feaLDA is as follows:

- For each class label $k \in \{1, ..., K\}$
  - For each topic $j \in \{1, ..., T\}$, draw $\boldsymbol{\varphi}_{kj} \sim \text{Dir}(\boldsymbol{\beta}_{kj})$
- For each document $d \in \{1, ..., D\}$,
  - draw $\boldsymbol{\pi}_d \sim \text{Dir}(\boldsymbol{\gamma} \times \boldsymbol{\epsilon}_d)$
  - For each class label k, draw $\boldsymbol{\theta}_{d,k} \sim \text{Dir}(\boldsymbol{\alpha}_k)$
- For each word w_i in document d
  - Draw a class label $c_i \sim \text{Mult}(\boldsymbol{\pi}_d)$
  - Draw a topic $z_i \sim \text{Mult}(\boldsymbol{\theta}_{d,c_i})$
  - Draw a word $w_i \sim \text{Mult}(\boldsymbol{\varphi}_{c_i,z_i})$

First, one draws a class label c from the per-document class label proportion $\boldsymbol{\pi}_d$. Following that, one draws a topic z from the per-document topic proportion $\boldsymbol{\theta}_{d,c}$ conditioned on the sampled class label c. Finally, one draws a word from the per-corpus word distribution $\boldsymbol{\varphi}_{z,c}$ conditioned on both topic z and class label c.

It is worth noting that if we assume that the class distribution $\boldsymbol{\pi}$ of the training data is observed and the number of topics is set to 1, then our feaLDA model is reduced to labeled LDA [19] where during training, words can only be assigned to the observed class labels in the document. If we allow multiple topics to be modelled under each class label, but don't incorporate the labelled feature constraints, then our feaLDA model is reduced to pLDA [20]. Both labelled LDA and pLDA actually imply a different generative process where class distribution for each document is observed, whereas our feaLDA model incorporates supervised information in a more principled way by introducing the transformation matrices $\boldsymbol{\lambda}$ and $\boldsymbol{\epsilon}$ for encoding the prior knowledge derived from both document labels and labelled features to modify the Dirichlet priors of document specific class distributions and topic-word distributions. A detailed discussion on how this can be done is presented subsequently.

3.1 Incorporating Supervised Information

Incorporating Document Class Labels: feaLDA incorporates the supervised information from document class labels by constraining that a training document can only be generated from the topic set with class labels correspond to the document's observed label set. This is achieved by introducing a dependency link from the document label matrix $\boldsymbol{\epsilon}$ to the Dirichlet prior $\boldsymbol{\gamma}$. Suppose a corpus has 2 unique labels denoted by $\boldsymbol{C} = \{c_1, c_2\}$ and for each label c_k there are 5 topics denoted by $\boldsymbol{\theta}_{c_k} = \{z_{1,c_k}, ...z_{5,c_k}\}$. Given document d's observed label vector $\boldsymbol{\epsilon}_d = \{1, 0\}$ which indicates that d is associated with class label c_1, we can encode the label information into feaLDA as

$$\boldsymbol{\gamma}_d = \boldsymbol{\epsilon}_d^T \times \boldsymbol{\gamma}. \tag{1}$$

where $\boldsymbol{\gamma} = \{\gamma_1, \gamma_2\}$ is the Dirichlet prior for the per-document class proportion $\boldsymbol{\pi}_d$ and $\boldsymbol{\gamma}_d = \{\gamma_1, 0\}$ is the modified Dirichlet prior for document d after encoding the class label information. This ensures that d can only be generated from topics associated with class label c_1 restricted by γ_1.

Incorporating Labelled Features: The second type of supervision that feaLDA accommodates is the labelled features automatically learned from the training data. This is motivated by the observation that LDA and existing supervised topic models usually set the Dirichlet prior of topic word distribution $\boldsymbol{\beta}$ to a symmetric value, which assumes each term in the corpus vocabulary is equally important before having observed any actual data. However, from a classification point of view, this is clearly not the case. For instance, words such as "endpoint", "delete" and "post" are more likely to appear in Web API documentations, whereas words like "money", "shop" and "chart" are more related to a document describing shopping. Hence, some words are more important to discriminate one class from the others. Therefore, we hypothesise that the word-class association probabilities or labelled features could be incorporated into model learning and potentially improve the model classification performance.

We encode the labelled features into feaLDA by adding an additional dependency link of $\boldsymbol{\varphi}$ (i.e., the topic-word distribution) on the word-class association probability matrix $\boldsymbol{\lambda}$ with dimension $C \times V'$, where V' denotes the labelled feature size and $V' <= V$. For word w_i, its class association probability vector is $\boldsymbol{\lambda}_{w_i} = (\lambda_{c_1,w_i}, ..., \lambda_{c_K,w_i})$, where $\sum_{c_k=1}^{K} \lambda_{c_k,w_i} = 1$. For example, the word "delete" in the API dataset with index w_t in the vocabulary has a corresponding class association probability vector $\lambda_{w_t} = (0.3, 0.7)$, indicating that "delete" has a probability of 0.3 associating with the non-API class and a probability of 0.7 with the API class. For each $w \in V$, if $w \in V'$, we can then incorporate labelled features into feaLDA by setting $\beta_{cw} = \lambda_{cw}$, otherwise the corresponding component of β is kept unchanged. In this way, feaLDA can ensure that labelled features such as "delete" have higher probability of being drawn from topics associated with the API class.

3.2 Model Inference

From the feaLDA graphical model depicted in Figure 2b, we can write the joint distribution of all observed and hidden variables which can be factored into three terms:

$$P(\mathbf{w},\mathbf{z},\mathbf{c}) = P(\mathbf{w}|\mathbf{z},\mathbf{c})P(\mathbf{z},\mathbf{c}) = P(\mathbf{w}|\mathbf{z},\mathbf{c})P(\mathbf{z}|\mathbf{c})P(\mathbf{c}) \tag{2}$$

$$= \int P(\mathbf{w}|\mathbf{z},\mathbf{c},\boldsymbol{\Phi})P(\boldsymbol{\Phi}|\boldsymbol{\beta})\,d\boldsymbol{\Phi} \cdot \int P(\mathbf{z}|\mathbf{c},\boldsymbol{\Theta})\,P(\boldsymbol{\Theta}|\boldsymbol{\alpha})\,d\boldsymbol{\Theta} \cdot \int P(\mathbf{c}|\boldsymbol{\Pi})\,P(\boldsymbol{\Pi}|\gamma)\,d\boldsymbol{\Pi}. \tag{3}$$

By integrating out $\boldsymbol{\Phi}$, $\boldsymbol{\theta}$ and $\boldsymbol{\Pi}$ in the first, second and third term of Equation 3 respectively, we can obtain

$$P(\mathbf{w}|\mathbf{z},\mathbf{c}) = \left(\frac{\Gamma(\sum_{i=1}^{V}\beta_{k,j,i})}{\prod_{i=1}^{V}\Gamma(\beta_{k,j,i})}\right)^{C\times T} \prod_k \prod_j \frac{\prod_i \Gamma(N_{k,j,i}+\beta_{k,j,i})}{\Gamma(N_{k,j}+\sum_i \beta_{k,j,i})} \tag{4}$$

$$P(\mathbf{z}|\mathbf{c}) = \left(\frac{\Gamma(\sum_{j=1}^{T}\alpha_{k,j})}{\prod_{j=1}^{T}\Gamma(\alpha_{k,j})}\right)^{D\times C} \prod_d \prod_k \frac{\prod_j \Gamma(N_{d,k,j}+\alpha_{k,j})}{\Gamma(N_{d,k}+\sum_j \alpha_{k,j})} \tag{5}$$

$$P(\mathbf{c}) = \left(\frac{\Gamma(\sum_{k=1}^{C}\gamma_k)}{\prod_{k=1}^{C}\Gamma(\gamma_k)}\right)^{D} \prod_d \frac{\prod_k \Gamma(N_{d,k}+\gamma_k)}{\Gamma(N_d+\sum_k \gamma_k)}, \tag{6}$$

where $N_{k,j,i}$ is the number of times word i appeared in topic j with class label k, $N_{k,j}$ is the number of times words are assigned to topic j and class label k, $N_{d,k,j}$ is the number of times a word from document d is associated with topic j and class label k, $N_{d,k}$ is the number of times class label k is assigned to some word tokens in document d, N_d is the total number of words in document d and Γ is the gamma function.

The main objective of inference in feaLDA is then to find a set of model parameters that can best explain the observed data, namely, the per-document class proportion $\boldsymbol{\pi}$, the per-document class label specific topic proportion $\boldsymbol{\theta}$, and the per-corpus word distribution $\boldsymbol{\varphi}$. To compute these target distributions, we need to know the posterior distribution $P(\boldsymbol{z},\boldsymbol{c}|\boldsymbol{w})$, i.e., the assignments of topic and class labels to the word tokens. However, exact inference in feaLDA is intractable, so we appeal to Gibbs sampler to approximate the posterior based on the full conditional distribution for a word token.

For a word token at position t, its full conditional distribution can be written as $P(z_t=j, c_t=k|\mathbf{w},\mathbf{z}^{-t},\mathbf{c}^{-t},\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\gamma})$, where $\mathbf{z}^{-t}$ and $\mathbf{c}^{-t}$ are vectors of assignments of topics and class labels for all the words in the collection except for the word at position t in document d. By evaluating the model joint distribution in Equation 3, we can yield the full conditional distribution as follows

$$P(z_t=j, c_t=k|\mathbf{w},\mathbf{z}^{-t},\mathbf{c}^{-t},\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\gamma}) \propto \frac{N_{k,j,w_t}^{-t}+\beta_{k,j,t}}{N_{k,j}^{-t}+\sum_i \beta_{k,j,i}} \cdot \frac{N_{d,k,j}^{-t}+\alpha_{k,j}}{N_{d,k}^{-t}+\sum_j \alpha_{k,j}} \cdot \frac{N_{d,k}^{-t}+\gamma_k}{N_d^{-t}+\sum_k \gamma_k}. \tag{7}$$

Table 1. Web APIs dataset statistics

Num. of Documents	Corpus size	Vocab. size	Avg. doc. length
1,547	1,096,245	35,427	708

Using Equation 7, the Gibbs sampling procedure can be run until a stationary state of the Markov chain has been reached. Samples obtained from the Markov chain are then used to estimate the model parameters according to the expectation of Dirichlet distribution, yielding the approximated per-corpus topic word distribution $\varphi_{k,j,i} = \frac{N_{k,j,i}+\beta_{k,j,i}}{N_{k,j}+\sum_i \beta_{k,j,i}}$, the approximated per-document class label specific topic proportion $\theta_{d,k,j} = \frac{N_{d,k,j}+\alpha_{k,j}}{N_{d,k}+\sum_j \alpha_{k,j}}$, and finally the approximated per-document class label distribution $\pi_{d,k} = \frac{N_{d,k}+\gamma_k}{N_d+\sum_k \gamma_k}$.

3.3 Hyperparameter Settings

For the feaLDA model hyperparameters, we estimate $\boldsymbol{\alpha}$ from data using maximum-likelihood estimation and fix the values of $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$.

Setting. $\boldsymbol{\alpha}$ A common practice for topic model implementation is to use symmetric Dirichlet hyperparameters. However, it was reported that using an asymmetric Dirichlet prior over the per-document topic proportions has substantial advantages over a symmetric prior [24]. We initialise the asymmetric $\boldsymbol{\alpha} = (0.1 \times L)/(K \times T)$, where L is the average document length and the value of 0.1 on average allocates 10% of probability mass for mixing. Afterwards for every 25 Gibbs sampling iterations, $\boldsymbol{\alpha}$ is learned directly from data using maximum-likelihood estimation [14,24]

$$\Psi(\alpha_{c,z}) = \Psi(\sum_{z=1}^{T} \alpha_{c,z}^{old}) + \log \bar{\boldsymbol{\theta}}_{c,z}, \tag{8}$$

where $\log \bar{\boldsymbol{\theta}}_{c,z} = \frac{1}{D}\sum_{d=1}^{D} \log \theta_{d,c,z}$ and Ψ is the digamma function.

Setting. $\boldsymbol{\beta}$ The Dirichlet prior $\boldsymbol{\beta}$ is first initialised with a symmetric value of 0.01 [23], and then modified by a transformation matrix $\boldsymbol{\lambda}$ which encodes the supervised information from the labelled feature learned from the training data.

Setting. $\boldsymbol{\gamma}$ We initialise the Dirichlet prior $\gamma = (0.1 \times L)/K$, and then modify it by the document label matrix $\boldsymbol{\epsilon}$.

4 Experimental Setup

The Web APIs Dataset. We evaluate the feaLDA model on the Web APIs dataset by crawling the Web pages from the API Home URLs of 1,553 Web APIs registered in ProgrammableWeb. After discarding the URLs which are out of date, we end up with 1,547 Web pages, out of which 622 Web pages are Web API documentations and the remaining 925 Web pages are not Web API documentations.

Preprocessing. The original dataset is in the HTML format. In the preprocessing, we first clean up the HTML pages using the HTML Tidy Library[3] to fix any mistakes in the Web page source. An HTML parser is subsequently used to extract contents from the HTML pages by discarding tags and the contents associating with the `<\script>` tag as these scripts are not relevant to classification. In the second step, we further remove wildcards, word tokens with non-alphanumeric characters and lower-case all word tokens in the dataset, followed by stop word removal and Porter stemming. The dataset statistics are summarised in Table 1.

Classifying a Document. In the feaLDA model, the class label of a test document is determined based on $P(\mathbf{c}|d)$, i.e., the probability of a class label given a document as specified in the per-document class label proportion $\boldsymbol{\pi}_d$. So given a learned model, we classify a document d by $\hat{c}_k = \text{argmax}_{c_k} P(c_k|d)$.

5 Experimental Results

In this section, we present the classification results of feaLDA on classifying a Web page as positive class (API documentation) or negative class (not API documentation) and compare against three supervised baselines, naive Bayes (NB), maximum entropy (MaxEnt), and Support Vector Machines (SVMs). We also evaluate the impact of incorporating labelled features on the classification performance by varying the proportion of labelled features used. Finally we compare feaLDA with some of the existing supervised topic models. All the results reported here are averaged over 5 trials where for each trial the dataset was randomly split into 80-20 for training and testing. We train feaLDA with a total number of 1000 Gibbs sampling iterations.

5.1 feaLDA Classification Results without Labelled Features

As the Web APIs dataset only contains two classes, positive or negative, we set the class number $K = 2$ in feaLDA. In this section, we only incorporate supervised information from the document class labels of the training set. In order to explore how feaLDA behaves with different topic settings, we experimented with topic number $T \in \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 15, 20\}$. It is worth noting that in feaLDA there are T topics associated with each class label. So for a setting of 2 class labels and 5 topics, feaLDA essentially models a total number of 10 topic mixtures.

Figure 3 shows the classification accuracy of feaLDA and three supervised baselines, namely, NB, MaxEnt and SVM. As can be seen from the figure, all the three supervised baselines achieve around 79% accuracy, with maxEnt giving a slightly higher accuracy of 79.3%. By incorporating the same supervision from document class labels, feaLDA outperforms all the three strong supervised baselines, giving the best accuracy of 80.5% at $T = 2$.

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

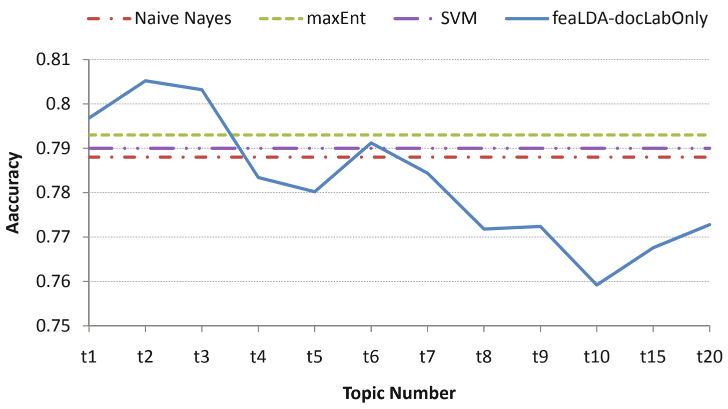


Fig. 3. feaLDA classification accuracy vs. different number of topics by incorporating supervision from class labels only

In terms of the impact of topic number on the model performance, it is observed that feaLDA performed the best around the topic setting $T = \{2, 3\}$. The classification accuracy drops and slightly fluctuates as the topic number increases. When the topic number is set to 1, feaLDA essentially becomes the labelled LDA model with two labelled topics being modelled corresponding to the two class labels. We see that the single topic setting actually yields worse result (i.e., 79.6% accuracy) than multiple topic settings, which shows the effectiveness of feaLDA over labelled LDA.

5.2 feaLDA Classification Results Incorporating Labelled Features

While feaLDA can achieve competitive performance by incorporating supervision from document labels alone, we additionally incorporated supervision from labelled features to evaluate whether a further gain in performance can be achieved. We extracted labelled features from the training data using information gain and discarded the features which have equal probability of both classes, resulting in a total of 29,000 features. In this experiment, we ran the feaLDA model with $T \in \{1, 2, 3, 4, 5\}$ as previous results show that large topic numbers do not yield good performance.

As observed in Figure 4, after incorporating both the document labels and labelled features, feaDLA has an substantial improvement over the model incorporating document labels only, regardless of the topic number setting. Particularly, feaLDA gives the best accuracy of 81.8% at $T = 3$, a clear 2.5% improvement over the best supervised baseline. It is also noted that when topic number is relatively large (i.e. $T = \{4, 5\}$), a significant performance drop is observed for feaLDA which only incorporates document labels; whereas feaLDA is less sensitive to topic number setting and can give fairly stable performance.

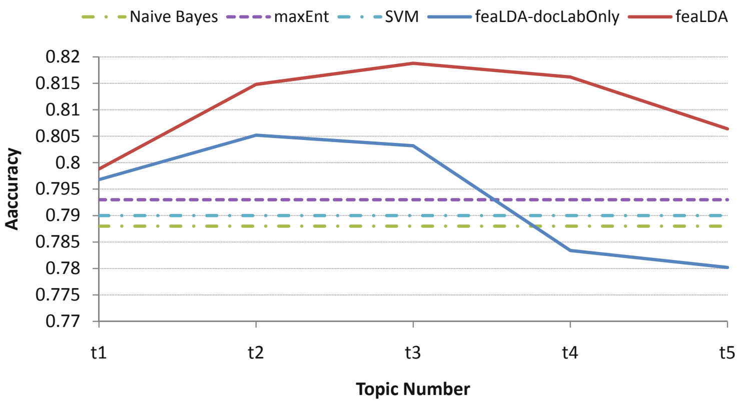


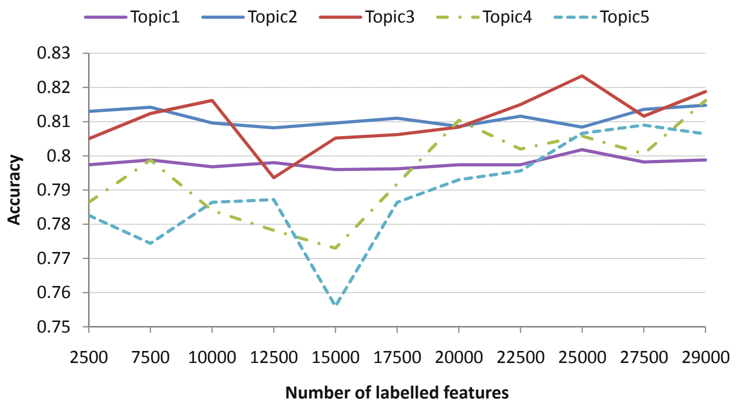
Fig. 4. feaLDA classification accuracy vs. different number of topics by incorporating supervision from both document class labels and labelled features.

5.3 feaLDA Performance vs. Different Feature Selection Strategies

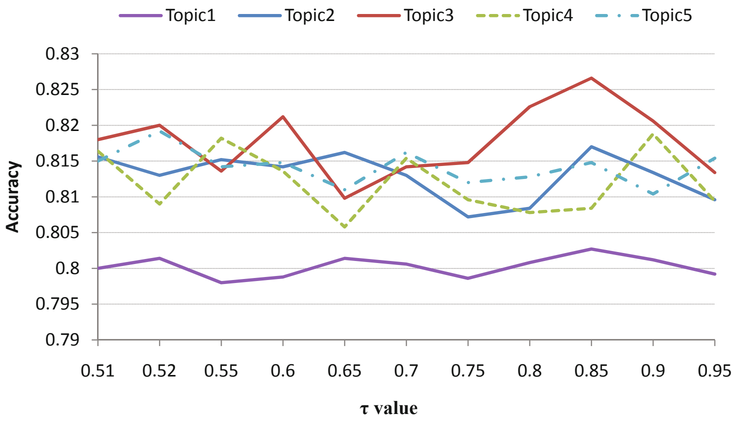
In the previous section, we directly incorporated all the labelled features into the feaLDA model. We hypothesise that using appropriate feature selection strategies to incorporate the most informative feature subset may further boost the model performance. In this section, we explore two feature selection strategies: (1) incorporate the top M features based on their information gain values; and (2) incorporate feature f if its highest class association probability is greater than a predefined threshold τ, i.e, $\text{argmax}_{c_k} P(c_k|f) > \tau$.

Figure 5a shows the classification accuracy of feaLDA by incorporating different number of most informative labelled features ranked by the information gain values. With topic setting $T = \{1, 2\}$, classification accuracy is fairly stable regardless of the number of features selected. However, with larger number of topics, incorporating more labelled features generally yields better classification accuracy. feaLDA with 3 topics achieves the best accuracy of 82.3% by incorporating the top 25,000 features, slightly outperforming the model with all features incorporated by 0.5%.

On the other hand, incorporating labelled features filtered by some predefined threshold could also result in the improvement of classification performance. As can be seen from Figure 5b, similar accuracy curves are observed for feaLDA with topic setting $T = \{1, 2, 3\}$, where they all achieved the best performance when $\tau = 0.85$. Setting higher threshold value, i.e. beyond 0.85, results in performance drop for most of the models as the number of filtered features becomes relatively small. In consistent with the previous results, feaLDA with 3 topics still outperforms the other topic settings giving the best accuracy of 82.7%, about 1% higher than the result incorporating all the features and 3.4% higher than the best supervised baseline model MaxEnt. From the above observations, we conclude that 3 topics and a feature-class association threshold $\tau = 0.85$ are the optimal model settings for feaLDA in our experiments.



(a) feaLDA classification accuracy vs. different number of features.



(b) feaLDA classification accuracy vs. different feature class probability threshold τ.

Fig. 5. feaLDA performance vs. different feature selection strategies

5.4 Comparing feaLDA with Existing Supervised Topic Models

In this section, we compare the overall performance of feaLDA with two supervised topic models (i.e. labelled LDA and pLDA) as well as three supervised baseline models on the APIs dataset. Apart from classification accuracy, we also report the recall, precision and F1 score for the positive class (true API label), which are summarised in Table 2.

It can be seen from Table 2 that although both feaLDA and labeled LDA give similar precision values, feaLDA outperforms labeled LDA in recall by almost 10%. Overall, feaLDA significantly outperforms labeled LDA by 6% in F1 score and 3% in accuracy. While labeled LDA simply defines a one-to-one correspondence between LDA's latent topics and document labels, pLDA extended labelled LDA by allowing multiple topics being modelled under each class label.

Table 2. Comparing feaLDA with existing supervised approaches. (Unit in %, numbers in bold face denote the best result in their respective row.)

	Naive Bayes	SVM	maxEnt	labeled LDA	pLDA	feaLDA
Recall	**79.2**	70.8	69.3	59.8	65.9	68.8
Precision	71.0	75.4	77.4	85.1	82.1	**85.2**
F1	74.8	73.1	73	70.2	73.1	**76**
Accuracy	78.6	79	79.3	79.8	80.5	**82.7**

Table 3. Topics extracted by feaLDA with $K = 2, T = 3$

Positive	T1: nbsp quot gt lt http api amp type code format valu json statu paramet element
	T2: lt gt id type http px com true url xml integ string fond color titl date
	T3: api http user get request url return string id data servic kei list page paramet
Negative	T1: px color font background pad margin left imag size border width height text div thread
	T2: servic api site develop data web user applic http get amp email contact support custom
	T3: obj park flight min type citi air fizbber airlin stream school die content airport garag

Although pLDA (with the optimal topic setting $T = 2$) improves upon labeled LDA, it is still worse than feaLDA with its F-measure nearly 3% lower and accuracy over 2% lower compared to feaLDA. This demonstrates the effectiveness of feaLDA in incorporating labelled features learned from the training data into model learning.

When compared to the supervised baseline models, feaLDA outperforms the supervised baselines in all types of performance measure except recall. Here we would like to emphasise that one of our goals is to develop a Web APIs discovery engine on the Web scale. So considering the fact that the majority of the Web pages are not related to Web API documentation, applying a classifier such as feaLDA that can offer high precision while maintaining reasonable recall is crucial to our application.

5.5 Topic Extraction

Finally, we show some topic examples extracted by feaLDA with 2 class label and 3 topics. As listed in Table 3, the 3 topics in the top half of the table were generated from the positive API class and the remaining topics were generated from the negative class, with each topic represented by the top 15 topic words.

By inspecting the topics extracted by feaLDA, it is revealed that, most of the words appear in the topics with true API label (positive class) are fairly technical such as *json*, *statu*, *paramet*, *element*, *valu*, *request* and *string*, etc. In contrast, topics under the negative class contain many words that are not likely to appear in an API documentation, such as *contact*, *support*, *custom*, *flight*, *school*, etc. This illustrates the effectiveness of feaLDA in extracting class-associated topics from text, which can potentially be used for Web service annotation in the future extension of our search engine.

6 Conclusions

In this paper, we presented a supervised topic model called feature LDA (feaLDA) which offers a generic framework for text classification. While most of the supervised topic models [2,19,20] can only encode supervision from document labels for model learning, feaLDA is capable to incorporate two different types of supervision from both document label and labelled features for effectively improving classification performance. Specifically, the labelled features can be learned automatically from training data and are used to constrain the asymmetric Dirichlet prior of topic distributions. Results from extensive experiments show that, the proposed feaLDA model significantly outperforms three strong supervised baselines (i.e. NB, SVM and MaxEnt) as well as two closely related supervised topic models (i.e. labeled LDA and pLDA) for more than 3% in accuracy. More importantly, feaLDA offers very high precision performance (more than 85%), which is crucial to our Web APIs search engine to maintain a low false positive rate as majority pages on the Web are not related to APIs documentation.

In the future, we plan to develop a self-training framework where unseen data labelled with high confidence by feaLDA are added to the training pool for iteratively retraining the feaLDA model with potentially further performance improvement. Another direction we would like to pursue is to extend feaLDA for multiple class classification and evaluate it on datasets from different domains.

Acknowledgements. This work was partly funded by the EU project VPH-Share (FP7-269978).

References

1. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent Dirichlet allocation. J. Mach. Learn. Res. 3, 993–1022 (2003)
2. Blei, D.M., McAuliffe, J.D.: Supervised topic models. Arxiv preprint arXiv:1003.0783 (2010)
3. Erl, T.: SOA Principles of Service Design. The Prentice Hall Service-Oriented Computing Series. Prentice Hall (2007)
4. Erosheva, E., Fienberg, S., Lafferty, J.: Mixed-membership models of scientific publications. Proceedings of the National Academy of Sciences of the United States of America 101(suppl. 1), 5220 (2004)
5. Gomadam, K., Ranabahu, A., Nagarajan, M., Sheth, A.P., Verma, K.: A faceted classification based approach to search and rank web apis. In: Proceedings of ICWS, pp. 177–184 (2008)
6. Hadley, M.: Web Application Description Language. Member submission, W3C (2009)
7. Kopecký, J., Gomadam, K., Vitvar, T.: hRESTS: an HTML Microformat for Describing RESTful Web Services. In: Proceedings of the International Conference on Web Intelligence (2008)
8. Lacoste-Julien, S., Sha, F., Jordan, M.I.: DiscLDA: Discriminative learning for dimensionality reduction and classification. In: NIPS, vol. 21 (2008)

9. Lin, C., He, Y.: Joint sentiment/topic model for sentiment analysis. In: Proceedings of Conference on Information and Knowledge Management, CIKM (2009)
10. Lin, C., He, Y., Everson, R., Rüger, S.: Weakly-Supervised Joint Sentiment-Topic Detection from Text. IEEE Transactions on Knowledge and Data Engineering, TKDE (2011)
11. Maleshkova, M., Pedrinaci, C., Domingue, J.: Investigating web apis on the world wide web. In: European Conference on Web Services, pp. 107–114 (2010)
12. McCallum, A., Corrada-Emmanuel, A., Wang, X.: Topic and role discovery in social networks. In: Proceedings of IJCAI, pp. 786–791 (2005)
13. Mimno, D., McCallum, A.: Topic models conditioned on arbitrary features with dirichlet-multinomial regression. In: Uncertainty in Artificial Intelligence, pp. 411–418. Citeseer (2008)
14. Minka, T.: Estimating a Dirichlet distribution. Technical report, MIT (2003)
15. Pedrinaci, C., Domingue, J.: Toward the next wave of services: linked services for the web of data. Journal of Universal Computer Science 16(13), 1694–1719 (2010)
16. Pedrinaci, C., Liu, D., Lin, C., Domingue, J.: Harnessing the crowds for automating the identification of web apis. In: Intelligent Web Services Meet Social Computing at AAAI Spring Symposium (2012)
17. Pedrinaci, C., Domingue, J., Sheth, A.: Semantic Web Services. In: Handbook on Semantic Web Technologies. Springer (2010)
18. Pilioura, T., Tsalgatidou, A.: Unified Publication and Discovery of Semantic Web Services. ACM Trans. Web 3(3), 1–44 (2009)
19. Ramage, D., Hall, D., Nallapati, R., Manning, C.D.: Labeled LDA: A supervised topic model for credit attribution in multi-labeled corpora. In: Proceedings of EMNLP, pp. 248–256 (2009)
20. Ramage, D., Manning, C.D., Dumais, S.: Partially labeled topic models for interpretable text mining. In: Proceedings of KDD, pp. 457–465 (2011)
21. Rosen-Zvi, M., Griffiths, T., Steyvers, M., Smyth, P.: The author-topic model for authors and documents. In: Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI), pp. 487–494 (2004)
22. Steinmetz, N., Lausen, H., Brunner, M.: Web Service Search on Large Scale. In: Baresi, L., Chi, C.-H., Suzuki, J. (eds.) ICSOC-ServiceWave 2009. LNCS, vol. 5900, pp. 437–444. Springer, Heidelberg (2009)
23. Steyvers, M., Griffiths, T.: Probabilistic topic models. In: Handbook of Latent Semantic Analysis, vol. 427 (2007)
24. Wallach, H., Mimno, D., McCallum, A.: Rethinking lda: Why priors matter, vol. 22, pp. 1973–1981 (2009)
25. Wang, X., Mohanty, N., McCallum, A.: Group and topic discovery from relations and text. In: Proceedings of Intl. Workshop on Link Discovery, pp. 28–35 (2005)

Efficient Execution of Top-K SPARQL Queries

Sara Magliacane[1,2], Alessandro Bozzon[1], and Emanuele Della Valle[1]

[1] Politecnico of Milano, P.za L. Da Vinci, 32. I-20133 Milano - Italy
[2] VU University Amsterdam, De Boelelaan 1081a, The Netherlands

Abstract. Top-k queries, i.e. queries returning the top k results ordered by a user-defined scoring function, are an important category of queries. Order is an important property of data that can be exploited to speed up query processing. State-of-the-art SPARQL engines underuse order, and top-k queries are mostly managed with a *materialize-then-sort* processing scheme that computes all the matching solutions (e.g. thousands) even if only a limited number k (e.g. ten) are requested. The $\mathcal{S}$PARQL-$\mathcal{R}$ANK algebra is an extended SPARQL algebra that treats order as a first class citizen, enabling efficient *split-and-interleave* processing schemes that can be adopted to improve the performance of top-k SPARQL queries. In this paper we propose an incremental execution model for $\mathcal{S}$PARQL-$\mathcal{R}$ANK queries, we compare the performance of alternative physical operators, and we propose a rank-aware join algorithm optimized for native RDF stores. Experiments conducted with an open source implementation of a $\mathcal{S}$PARQL-$\mathcal{R}$ANK query engine based on ARQ show that the evaluation of top-k queries can be sped up by orders of magnitude.

1 Introduction

As the adoption of SPARQL as a query language for Web data increases, practitioners are showing a growing interest in top-k queries [5,6], i.e. queries returning the top k results ordered by a specified scoring function. Simple top-k queries can be expressed in SPARQL 1.0 by including the ORDER BY and LIMIT clauses, which impose an order on the result set and limit the number of results. SPARQL 1.1 additionally enables the specification of complex scoring functions through the use of projection expressions that can define a variable to be used in the ORDER BY clause. Listing 1.1 provides an example SPARQL 1.1 top-k query that will be used as a running example.

```
SELECT ?product ?offer (g1(?avgRat1) + g2(?avgRat2) + g3(?price1) AS ?score)
WHERE { ?product hasAvgRating1 ?avgRat1 .
 ?product hasAvgRating2 ?avgRat2 .
 ?product hasName ?name .
 ?product hasOffers ?offer .
 ?offer hasPrice ?price1 }
ORDER BY DESC(?score) LIMIT 10
```

Listing 1.1. A top-k SPARQL query that retrieves the best ten offers of products ordered by a function of user ratings and offer price; g_i are normalization functions and the bold letters represent the abbreviations used in the following examples

In most of the algebraic representation of SPARQL, the algebraic operators that evaluate the ORDER BY and LIMIT clauses are result modifiers, i.e. operators that alter the sequence of solution mappings *after* the full evaluation of the graph pattern in the WHERE clause. For instance, the query in Listing 1.1 is executed according to the plan in Fig. 1.a: solutions matching the WHERE clause are drawn from the storage until the whole result set is materialized; then, the project expression *?score* is evaluated by the EXTEND operator on each solution and used to order the result set; finally the top 10 results are returned.

This *materialize-then-sort* scheme can hinder the performance of SPARQL top-k queries, as a SPARQL engine might process thousands of matching solutions, compute the score for each of them and order the result set, even if only a limited number (e.g. ten) were requested. Moreover, the ranking criteria can be expensive to compute and, therefore, should be evaluated only when needed and on the minimum possible number of mappings. It has been shown [9] that query plans where the scoring function is evaluated in an incremental, non-blocking manner, access a lower number of mappings, thus yielding better performance. $\mathcal{S}$PARQL-$\mathcal{R}$ANK [1] is a rank-aware algebra for SPARQL that has been designed to address such a need, as it enables the definition of query plans as shown in Fig. 1.b, where the evaluation of the scoring function is split and delegated to rank-aware operators (the ranking operator ρ and the rank-join $\bowtie$) that are interleaved with other operators and incrementally order the mappings extracted from the data store.

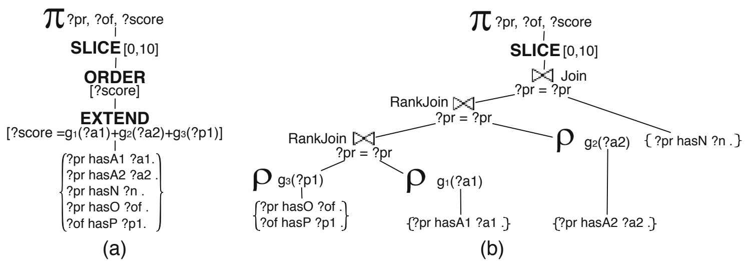


Fig. 1. The (a) standard and (b) $\mathcal{S}$PARQL-$\mathcal{R}$ANK algebraic query plan for the top-k SPARQL query in Listing 1.1

Contribution. In this paper, we propose an incremental execution model for top-k queries based on the $\mathcal{S}$PARQL-$\mathcal{R}$ANK algebra. We show how existing work on rank-aware physical operators can be adapted to enable the incremental execution of top-k SPARQL queries over RDF data stores. The focus of our work is on top-k SPARQL query evaluation at the *query engine level*, thus abstracting from the underlying data storage layer. Nonetheless, we show how the availability of dedicated storage data structures for sorted (retrieving mappings sorted by their score) or random access (retrieving mappings matching a join attribute value) can speed-up the execution of top-k queries. While random access is available in both in RDBMS and native RDF storage systems (although with better performance in the latter), sorted access is typically unavailable in RDF stores.

However, we show how the presence of sorted access can boost the performance of top-k SPARQL queries, and we elaborate on strategies to obtain it. Based on the results of a comparative analysis of state-of-the-art physical operators, we propose a rank-aware join algorithm (namely RSEQ) optimized for native RDF stores. We also describe our implementation experience based on ARQ 2.8.9[1] (namely ARQ-$\mathcal{R}$ANK). We provide experimental evidence that the incremental execution model described in the paper speeds up the execution of top-k queries in ARQ-$\mathcal{R}$ANK by orders of magnitude w.r.t. the original ARQ implementation.

Organization of the Paper. In Section 2, we discuss the related work. Section 3 reviews the $\mathcal{S}$PARQL-$\mathcal{R}$ANK algebra proposed in [1]. Section 4 introduces a set of incremental rank-aware physical operators, proposes a rank-join algorithm optimized for native RDF stores, discusses sorted access in RDF stores, and presents three rank-aware optimization techniques. Section 5 focuses on the evaluation based on an extended version of BSBM [3] and reports on the experiments with ARQ-$\mathcal{R}$ANK. Section 6 presents conclusions and future work.

2 Related Work

SPARQL query optimization is a consolidated field of research. Existing approaches focus on algebraic [11,13] or selectivity-based optimizations [14]. Despite an increasing need from practitioners [5,6], few works address top-k query optimization in SPARQL. In state-of-the-art query engines, a basic top-k optimization was introduced in ARQ 2.8.9 [5], but the ORDER BY and LIMIT clauses are still evaluated after the completion of the other operations and on the complete set of solutions. OWLIM and Virtuoso[2] have some basic ranking features that precompute the rank of most popular nodes in the RDF graph based on the graph link density. These ranks are used to order query results by popularity, not based on a user-specified scoring function. Moreover, the query execution is not incremental, even if Virtuoso uses an anytime algorithm.

Our work builds on the results of several well-established techniques for the efficient evaluation of top-k queries in relational databases such as [9,7,8,18,15] where efficient rank-aware operators are investigated, and [4] where a rank-aware relational algebra and the RankSQL DBMS are described. In particular, we considered the algorithms described in [7,8], while building on the discussions about data access types described in [15]. The application of such results to SPARQL is not straightforward, as SPARQL and relational algebra have equivalent expressive power, while just a subset of relational optimizations can be ported to SPARQL [13]. Moreover, relational rank-aware operators require dedicated data structures for sorted access, while they often do not assume to have data structures for random access. In contrast, sorted access is usually not available in native triplestores (as it is rare to have application-specific indexes), while random access is common, as required for the efficient evaluation of schema-free data

[1] Among many alternatives, we chose Jena ARQ, because of its neat design and the fact that it recently tackled the problem of top-k optimization [5].

[2] OWLIM: `http://bit.ly/N9ZRG3Virtuoso:http://bit.ly/SfRWKm`

such as RDF. In a previous work [1], we presented the $\mathcal{S}$PARQL-$\mathcal{R}$ANK algebra, and applied it to the execution of top-k SPARQL queries on top of virtual RDF stores through query rewriting over a rank-aware RDBMS. In this paper, we discuss and evaluate the execution of top-k SPARQL queries also in native RDF stores, offering an extensive discussion on rank-aware operators, optimization techniques, and their application to different data access configurations.

Several works extend the standard SPARQL algebra to allow the definition of ranking predicates [10,20]. AnQL [17] is an extension of the SPARQL language and algebra able to address a wide variety of queries (including top-k ones) over annotated RDF graphs; our approach, on the other hand, requires no annotations, and can be applied to any state-of-the-art SPARQL engine.

Straccia [19] describes an ontology mediated top-k information retrieval system over relational databases, where user queries are rewritten into a set of conjunctive queries, which are translated in SQL queries and executed on a rank-aware RDBMS [4]; the obtained results are merged into the final top-k answers. Another rank-join algorithm, the Horizon based Ranked Join, is introduced [21] and aims at optimizing twig queries on weighted data graphs. In this case, results are ranked based on the underlying cost model, not based on an ad-hoc scoring function as in our work. The SemRank system [12] uses a rank-join algorithm to calculate the top-k most relevant paths from all the paths that connect two resources specified in the query. However, the application context of this algorithm is different from the one we present, because it targets paths and ranks them by relevance using IR metrics, and the focus is not on query performance optimization.

Wagner et al. [2] introduce PBRJ, a top-k join algorithm for federated queries over Linked Data. The proposed processing model is push-based, i.e. operators push mappings to subsequent operators, and, given a set or pre-defined ranking criteria, assumes a-priori knowledge of the upper-bounds and lower-bounds for the scores contained in each data source. By leveraging these bounds it is possible to decide at execution-time which data source to query. The complete content of the involved sources is materialized locally and, if sorted access is not available, it is fully ordered. Due to domain-specific assumptions, PBRJ is able to define a better estimation of unseen join results than standard rank-join algorithms. This work is complementary to our approach, as our work bases on the traditional pull-based processing model, in which operators pull mappings from their input operators; we consider RDF data stored locally either in native or virtual RDF stores, and we make no assumption (e.g., upper- or lower-bound for scores and data access types) on the evaluated data sources. Data can be queried by user-defined ranking criteria, and efficient data access methods like random access (an important optimization factor in rank-join algorithms) can be exploited to further improve the query performance. We are currently investigating a hybrid processing model to address the federated query scenario, combining the strengths of both push and pull-based processing models.

3 Background

To support the following discussion, we review the existing formalization of SPARQL in [11] and our $\mathcal{S}$PARQL-$\mathcal{R}$ANK [1] algebra.

3.1 Basic SPARQL Definitions

In SPARQL, the WHERE clause contains a **graph pattern** that can be constructed using the OPTIONAL, UNION, FILTER and JOIN operators. Given three sets I, L and V (IRIs, literals and variables), a tuple from $(I \cup L \cup V) \times (I \cup V) \times (I \cup L \cup V)$ is a triple pattern. A **Basic Graph Pattern** (BGP) is a set of triple patterns connected by the JOIN operator.

The semantics of SPARQL is based on the notion of **mapping**, defined in [11] as a partial function $\mu : V \rightarrow (I \cup L \cup B)$, where B is the set of blank nodes. The domain of μ, denoted by *dom(μ)*, is the subset of V where μ is defined. Let P be a graph pattern, *var(P)* denotes the set of variables occurring in P. Given a triple pattern t and a mapping μ such that $var(t) \subseteq dom(\mu)$, $\mu(t)$ is the triple obtained by replacing the variables in t according to μ.

Using these definitions, it is possible [11][13] to define the semantics of SPARQL queries with an algebra having a set of operators – Selection (σ), Join ($\bowtie$), Union ($\cup$), Difference($\setminus$) and Left Join ($⟕$)– operating on **sets of mappings** denoted with Ω. The evaluation of a SPARQL query is based on its translation into an algebraic tree composed of those algebraic operators.

3.2 The $\mathcal{S}$PARQL-$\mathcal{R}$ANK Algebra

$\mathcal{S}$PARQL-$\mathcal{R}$ANK is a rank-aware framework for top-k queries in SPARQL, based on the $\mathcal{S}$PARQL-$\mathcal{R}$ANK algebra, an extension of the SPARQL algebra that supports ranking as a first-class construct. The central concept of the $\mathcal{S}$PARQL-$\mathcal{R}$ANK algebra is the *ranked set of mappings*, an extension of the standard SPARQL definition of a set of mappings that embeds the notion of ranking.

$\mathcal{S}$PARQL-$\mathcal{R}$ANK supports top-k queries in SPARQL with an ORDER BY clause that can be formulated as a scoring function combining several ranking criteria. Given a graph pattern P, a **ranking criterion** $b(?x_1, \ldots, ?x_m)$ is a function defined over a set of m variables $?x_j \in var(P)$. The evaluation of a ranking criterion on a mapping μ, indicated by $b[\mu]$, is the substitution of all of the variables $?x_j$ with the corresponding values from the mapping. A criterion b can be the result of the evaluation of any built-in function of query variables that ensures that $b[\mu]$ is a numerical value. We define as max_b the application-specific maximal possible value for the ranking criterion b.

A **scoring function** on P is an expression $\mathcal{F}(b_1, \ldots, b_n)$ defined over the set B of n ranking criteria. The evaluation of a scoring function $\mathcal{F}$ on a mapping μ, indicated by $\mathcal{F}[\mu]$, is the value of the function when all of the $b_i[\mu]$, where $\forall i = 1, \ldots, n$, are evaluated. As typical in ranking queries, the scoring function $\mathcal{F}$ is assumed to be **monotonic**, i.e., a $\mathcal{F}$ for which holds $\mathcal{F}(b_1[\mu_1], \ldots, b_n[\mu_1]) \geq \mathcal{F}(b_1[\mu_2], \ldots, b_n[\mu_2])$ when $\forall i : b_i[\mu_1] \geq b_i[\mu_2]$.

In order to evaluate the scoring function, all the variables in $var(P)$ that contribute in the evaluation of $\mathcal{F}$ must be bound. Since OPTIONAL and UNION clauses can introduce unbound variables, we assume all the variables in $var(P)$ to be *certain variables*, as defined in [13][3], i.e. variables that are certainly bound for every mapping produced by P. An extension of $\mathcal{S}$PARQL-$\mathcal{R}$ANK towards the relaxation of the *certain variables* constraint is part of the future work. Listing 1.1 provides an example of the scoring function $\mathcal{F}$ calculated over the ranking criteria g_1*(?avgRat1)*, g_2*(?avgRat2)*, and g_3*(?price1)*.

A key property of $\mathcal{S}$PARQL-$\mathcal{R}$ANK is the ability to retrieve the first k results of a top-k query before scanning the complete set of mappings resulting from the evaluation of the graph pattern. To enable such a property, the mappings progressively produced by each operator should flow in an order consistent with the final order, i.e., the order imposed by $\mathcal{F}$. When the evaluation of a SPARQL top-k query starts on the graph pattern the resulting mappings are unordered. As soon as a subset $\mathcal{B} \subseteq B$, s.t. $\mathcal{B} = \{b_1, \ldots, b_j\}$ (with $j \leq |B|$) of the ranking criteria can be computed (i.e., when $var(b_k) \subseteq dom(\mu)\ \forall k = 1, \ldots, j$), an order can be imposed to a set of mappings Ω by evaluating for each $\mu \in \Omega$ the ***upper-bound*** of $\mathcal{F}[\mu]$ as:

$$\overline{\mathcal{F}}_{\mathcal{B}}(b_1, \ldots, b_n)[\mu] = \mathcal{F}\begin{pmatrix} b_i = b_i[\mu] & \text{if } b_i \in \mathcal{B} & \\ b_i = max_{b_i} & \text{otherwise} & \forall i \end{pmatrix}$$

Note that if $\mathcal{B} = B$, then $\overline{\mathcal{F}}_{\mathcal{B}}[\mu] = \mathcal{F}_B[\mu]$. Therefore, it is clear that for any subset of ranking criteria $\mathcal{B}$, the value of $\overline{\mathcal{F}}_{\mathcal{B}}[\mu]$ is the upper-bound of the score that μ can obtain, when $\mathcal{F}_B[\mu]$ is completely evaluated, by assuming that all the ranking criteria still to evaluate will return their maximal possible value.

A **ranked set of mappings** $\Omega_{\mathcal{B}}$ w.r.t. a scoring function $\mathcal{F}$ and a set $\mathcal{B}$ of ranking criteria, is the set of mappings Ω augmented with an order relation $<_{\Omega_{\mathcal{B}}}$ defined over Ω, which orders mappings by their upper-bound scores, i.e., $\forall\ \mu_1,\ \mu_2 \in \Omega :\ \mu_1 <_{\Omega_{\mathcal{B}}} \mu_2 \iff \overline{\mathcal{F}}_{\mathcal{B}}[\mu_1] < \overline{\mathcal{F}}_{\mathcal{B}}[\mu_2]$. A set of mappings on which no ranking criteria is evaluated ($\mathcal{B} = \emptyset$) is consistently denoted as $\Omega_{\emptyset}$ or Ω.

The monotonicity of $\mathcal{F}$ implies that $\overline{\mathcal{F}}_{\mathcal{B}}$ is always an upper-bound of $\mathcal{F}$, i.e. $\overline{\mathcal{F}}_{\mathcal{B}}[\mu] \geq \mathcal{F}[\mu]$ for any mapping $\mu \in \Omega_{\mathcal{B}}$, thus guaranteeing that the order imposed by $\overline{\mathcal{F}}_{\mathcal{B}}$ is consistent with the order imposed by $\mathcal{F}$.

Algebraic Operators. The $\mathcal{S}$PARQL-$\mathcal{R}$ANK algebra introduces a new *rank operator* ρ, representing the evaluation of a single ranking criterion, and redefines the Selection (σ), Join ($\bowtie$), Union ($\cup$), Difference($\setminus$) and Left Join ($⟕$) operators, enabling them to process and output ranked sets of mappings. For the sake of brevity, we present ρ and $\bowtie$, referring the reader to [1] for further details.

The *rank operator* ρ_b evaluates the ranking criterion $b \in B$ upon a ranked set of mappings $\Omega_{\mathcal{B}}$ and returns $\Omega_{\mathcal{B} \cup \{b\}}$, i.e. the same set ordered by $\overline{\mathcal{F}}_{\mathcal{B} \cup \{b\}}$. Thus, by definition $\rho_b(\Omega_{\mathcal{B}}) = \Omega_{\mathcal{B} \cup \{b\}}$.

The extended $\bowtie$ operator has a standard semantics for the membership property [11], while it defines an order relation on its output mappings: given two ranked sets of mappings $\Omega^{'}_{B_1}$ and $\Omega^{''}_{B_2}$ ordered with respect to two sets of

[3] This can be ensured by an efficiently verifiable syntactical condition.

ranking criteria B_1 and B_2, the join $\Omega'_{B_1} \bowtie \Omega''_{B_2}$ produces a ranked set of mappings ordered by $\overline{\mathcal{F}}_{B_1 \cup B_2}$. Formally $\Omega'_{B_1} \bowtie \Omega''_{B_2} \equiv (\Omega' \bowtie \Omega'')_{B_1 \cup B_2}$.

Algebraic Equivalences. Query optimization relies on algebraic equivalences to produce several equivalent formulations of a query. The $\mathcal{S}$PARQL-$\mathcal{R}$ANK algebra defines a set of algebraic equivalences that take into account the order property. The rank operator ρ can be pushed-down to impose an order to a set of mappings; such order can be then exploited to limit the number of mappings flowing through the physical execution plan, while allowing the production of the k results. In the following, we focus on the equivalences that apply to the ρ and $\bowtie$ operators (see [1] for the complete set of equivalences):

1. **Rank splitting** $[\Omega_{\{b_1,b_2,\ldots,b_n\}} \equiv \rho_{b_1}(\rho_{b_2}(\ldots(\rho_{b_n}(\Omega))\ldots))]$: allows splitting the criteria of a scoring function into a series of rank operations $(\rho_{b_1}, \ldots, \rho_{b_n})$, thus enabling the individual processing of the ranking criteria.
2. **Rank commutative law** $[\rho_{b_1}(\rho_{b_2}(\Omega_B)) \equiv \rho_{b_2}(\rho_{b_1}(\Omega_B))]$: allows the commutativity of the ρ operand with itself, thus enabling query planning strategies that exploit optimal ordering of rank operators.
3. **Pushing ρ over** $\bowtie$ [if Ω'' does not contain any variable of the ranking criterion b, then $\rho_b(\Omega'_{B_1} \bowtie \Omega''_{B_2}) \equiv \rho_b(\Omega'_{B_1}) \bowtie \Omega''_{B_2}$; if both Ω' and Ω'' contain all variables of b, then $\rho_b(\Omega'_{B_1} \bowtie \Omega''_{B_2}) \equiv \rho_b(\Omega'_{B_1}) \bowtie \rho_b(\Omega''_{B_2})$]: this law handles swapping $\bowtie$ with ρ, thus allowing to push the rank operator only on the operands whose variables also appear in b.

The new algebraic laws lay the foundation for query optimization, as discussed in the following section.

4 Execution of Top-K SPARQL Queries

In common SPARQL query engines (e.g. Jena ARQ), a query execution plan is a tree of physical operators designed according to a pull-based processing model. During execution, mappings are extracted iteratively from the root operator, which, in turn, will draw from the child operators only the intermediate mappings needed to produce the output. The same applies for the child operators, recursively up to the evaluation of Basic Graph Patterns (BGPs) in the storage layer. The execution is incremental unless some blocking operator appears in the execution plan (e.g. the ORDER BY operator, which materializes all the intermediate results to order them).

The $\mathcal{S}$PARQL-$\mathcal{R}$ANK algebra briefly presented in Section 3.2 enables an execution model in which the blocking ordering operator can be split in several non-blocking rank operators. Using the algebraic equivalences, it becomes possible to push these rank operators inside the execution tree and evaluate the order for each ranking criterion incrementally. The final order of the results, i.e. the order of the scoring function, is ensured by the other rank-aware operators.

In this section, we describe the $\mathcal{S}$PARQL-$\mathcal{R}$ANK incremental execution model and the related physical operators; then, we report on our investigations on a rank-aware optimizer that leverages the algebraic equivalences of Section 3.2.

4.1 Incremental Rank-Aware Physical Operators

The $\mathcal{S}$PARQL-$\mathcal{R}$ANK execution model creates rank-aware query plans, i.e. trees of physical operators that incrementally output ranked sets of mappings according to their *upper-bounds*. The execution stops as soon as the requested number of mappings has been drawn from the root operator. A rank-aware execution model calls for rank-aware physical operators. Some of them are of trivial nature; for instance, the *selection* operator only filters solutions that do not satisfy the FILTER clause, thus guaranteeing the preservation of mappings order. Other operators, e.g. ρ, $\bowtie$, $⟕$ and $\cup$, require more complex algorithms.

Rank. The rank operator ρ can exploit rank aggregation algorithms. This class of algorithms orders, in an incremental manner, lists of objects by combining several partial scores (e.g. the score for each ranking criterion) into one final score (e.g.the scoring function). MPro [7] is a state-of-the-art rank aggregation algorithm, which requires sorted access on one of the ranking criteria to be fully effective. To fit the $\mathcal{S}$PARQL-$\mathcal{R}$ANK execution model we adopted MPro as a ρ_b operator that maintains a priority queue containing all the mappings drawn from its input. Within the queue these mappings are ordered by $\overline{\mathcal{F}}_{\mathcal{B}\cup\{b\}}$, where $\mathcal{B}$ is the set of already evaluated criteria on the input set of mappings, while b is the ranking criteria to be evaluated by ρ_b. The operator ρ_b cannot output immediately each drawn mapping, since one of the next mappings could obtain a higher score after evaluation. Instead, it outputs the top ranked mapping of the priority queue μ only when it draws from its input a mapping μ' such that $\overline{\mathcal{F}}_{\mathcal{B}\cup\{b\}}[\mu] \geq \overline{\mathcal{F}}_{B}[\mu']$.

Join. Depending on the ordering and access patterns of its inputs, a rank-aware $\bowtie$ operator can be implemented with several physical operators. In the simplest case when only the left input is a ranked set of mapping, a standard *index join* algorithm can be adopted, which maintains the order of the output mappings according to the one of the left operand. The presence of several indexes in native triplestores (e.g. s, p, o, spo, pos, osp) guarantees fast random access to triples, thus enabling an optimized $n - way$ joining strategy called *streaming index join*[4], which performs lookups on the indexes by substituting already bound variables from previous inputs.

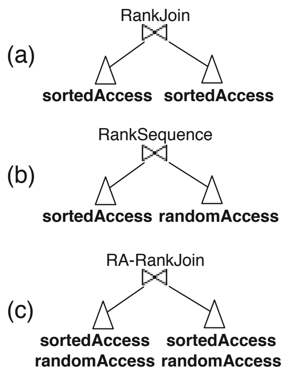


Fig. 2. Rank-join algorithms

Other input configurations require a rank-join algorithm. The idea of rank-join algorithms is to combine ranking and joining, by ordering progressively the results during the join operation. This can be achieved by taking advantage of the individual orders of its inputs to update, after each extraction from the inputs, an upper-bound of the scores of all join combinations not yet seen. A join result is returned only if it has a combined score greater than or equal to the upper-bound, thus ensuring that no other combination could possibly achieve a better

[4] The strategy is named *Sequence* in Jena ARQ, see http://bit.ly/O8e3Rm

score. In this work we consider three algorithms for rank-join (in Fig. 2(a-c)), each one characterized by a different configuration of access patterns on their left and right inputs.

HRJN [8] (and its variant HRJN*) is an example of rank-aware symmetrical hash join that has been shown to provide good performance. The basic HRJN and HRJN* operators assume *sorted access* (retrieving mappings sorted by their score) on both inputs. If the inputs offer *random access* (retrieving mappings matching a join attribute value), then some optimizations can be achieved. We will refer to the generalized version of HRJN endowed with both sorted and random access on both inputs as "RA-HRJN".

Given that efficient sorted access in not commonly available in native triple stores (Section 4.2 provides a detailed discussion on the topic), to exploit available random access mechanisms we propose **RankSequence** (RSEQ), a characterization of the HRJN rank-join template that requires minimum sorted access and leverages random access to improve performances. RSEQ is designed for supporting a configuration where one input provides only sorted access (S), and the other one supports only random access (R). To the best of our knowledge, no previous work on rank-join algorithms addresses these assumptions on data access[5]. The RSEQ algorithm supports a pull-based query model, for which the `getNext` method is presented in Figure 3. The algorithm requires the maintenance of a priority queue Q, which contains all the seen join combinations ordered by a scoring function f. In a rank-aware query plan, f is $\overline{\mathcal{F}}_{B_1 \cup B_2}$, where B_1 and B_2 are the sets of ranking criteria on the input ranked sets of mappings.

Algorithm 1 The RSEQ getNext method

```
S: sorted access input
R: random access input
$R_{top}$: maximal possible value for R
loop
  if Q is not empty then
    $topScore \leftarrow$ Q.topScore();
    if $topScore \geq$ Threshold then
      return Q.topMapping();
    end if
  end if
  $\mu_S \leftarrow$ S.getNext();
  probe R with $\mu_S$;
  for each retrieved join combination
    insert the join combination in Q;
  end for
  Threshold $\leftarrow$ f($\mu_S, R_{top}$);
end loop
```

Fig. 3. getNext method for *RSEQ*

The algorithm extracts a mapping μ_S from S and it probes R in order to get all corresponding join combinations, inserting them into the priority queue. Then it updates a threshold, which is the upper-bound of the scores for the not yet seen combinations. The mapping with the highest score in the priority queue is output only if it has a score greater or equal to the threshold.

LeftJoin. As all the variables in a $\mathcal{S}$PARQL-$\mathcal{R}$ANK query are assumed to be *certain variables*, all $⟕$ operators can have a ranked set of mapping only as left input. Thus, standard *index left-join* algorithms can be adopted, as they will output mappings in the same order of evaluation.

4.2 Sorted Access in Triplestores

The selection of the most suited rank-join operator is conditioned by the availability of sorted access or random access for its inputs. While random access on

[5] Similar assumptions are made by Upper [16], which is a *rank aggregation* algorithm, i.e. it is designed for the ρ operator.

triples is the basic operation in native triplestores, sorted access is not typically available at storage level. The ρ operators could be adopted to provide an initial order (according to one ranking criterion) to mappings extracted from the storage layer; however, such a solution is inherently inefficient, as it requires to process all the matching mappings. In the following, we show two cases in which we exploit or extend the design of the native triplestores to provide an efficient sorted access on triples and BGPs.

Exploiting Existing Indexes. Given a triple pattern of the form *(?s p ?o)*, in which p is a given predicate, if a ranking criterion is defined as the variable $?o$ that assumes literal values, then the triplestore native indexes can be exploited. Since triplestores usually provide POS B+ trees, where objects are ordered lexicographically, triples are already stored ordered by the variable value and, therefore, are extracted in the right order. Note that lexicographical order for numerical values means they will be retrieved in ascending order. The same effect can be obtained by reordering the set of triple patterns that compose a BGP so to position the triple pattern involved in the ranking criterion as first.

Creating Custom Indexes. If a ranking criterion involves the evaluation of an arbitrary function of variable values, or if the literal values should be ordered in descending order, then the native POS index cannot be exploited. Therefore, sorted access can be provided at storage level only by creating custom indexes, which store the evaluations of triples and/or BGPs in the order enforced by one or more ranking criterion. POS indexes can be still exploited by materializing the values of the arbitrary ranking criterion as attributes in the dataset.

4.3 Rank-aware Optimization Techniques

The goal of query optimization is the selection of an efficient execution plan for a given query. Many optimization techniques exist for SPARQL[6], but the addition of the *ranking* logical property brings novel optimization dimensions.

Several works on top-k query processing in RDBMSs propose optimization techniques that attempt to provide a (sub) optimal scheduling of rank [7] or rank-join operators [8] via dataset sampling or ranking selectivity estimation. An optimizer using both operators is presented in [4], where the cost of the generated plans is estimated by executing the plan on a sample of the dataset.

Unfortunately, previous works on top-k query planning in RDBMS cannot be directly ported to SPARQL engines, as data in an RDF storage can be "schema-free"; moreover, in some systems, it is possible to push the evaluations of BGPs down to the storage system, which can be optimized w.r.t. to a standard join by several techniques, like selectivity estimation optimization in native triplestores or SQL rewriting in the case of virtual RDF stores.

The design of a query planner for top-k queries demands for a detailed evaluation of the possible rank-aware configurations that might arise, a topic that we leave to future work. In this paper, we focus on the *rank* and *rank-join* operators and we discuss their application according to several optimization dimensions.

[6] Refer to [13] for a comprehensive review.

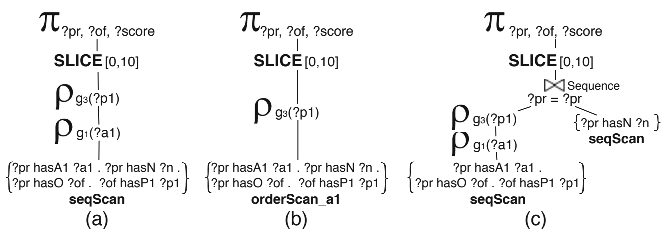


Fig. 4. Three examples of plans for the query in Listing 1.1: (a) ROB strategy with sequential access, (b) ROB strategy with sorted access on $a1$ (c) INTER strategy

Although simple, all of the proposed strategies proved effective in creating plans that considerably reduce the execution time of top-k queries and, therefore, could be easily adopted for heuristic-based query plan selection.

Ranking of BGP Strategy. The Ranking of BGP (ROB) strategy is a naïve planning technique that only uses rank operators. The evaluation of the BGP is pushed to the storage system, from which the mappings are fetched incrementally through several pipelined rank operators, one per each ranking criterion. The strategy requires little planning overhead, as it exploits only the algebraic transformations on rank operators.

Let P be the graph pattern of a query with a scoring function composed by n ranking criteria, the initial query plan is in the form: $\rho_{\{b_1,b_2,\ldots,b_n\}}(\Omega^P_\emptyset)$, where $\Omega^P_\emptyset$ is the unordered set of mapping resulting from evaluation of P. The rank-aware query plan, produced by the ROB strategy applying the splitting law for ρ, is in the form: $\rho_{b_1}(\rho_{b_2}(\ldots(\rho_{b_n}(\Omega^P_\emptyset))))$.

Fig. 4 depicts two example plans for the case study query, where the BGP evaluation is respectively performed with sequential access (Fig. 4.a) and sorted access on the variable $a1$ (Fig. 4.b). If the data source already provides *sorted access* according to one ranking criterion b_i, then the planner can reorder the ρ operators using the rank commutative law to have b_i evaluated first. Since the retrieved set of mappings is not $\Omega^P_\emptyset$, but $\Omega^P_{b_i}$, it can remove the corresponding ρ_{b_i} operator from the plan.

Interleaved Strategy. The ROB strategy can be extended to include an additional planning phase that splits the triple patterns containing variables of the ranking criteria ("ranked triple patterns") from the others. This approach is called Interleaved strategy (INTER), because it interleaves rank-aware with non-rank-aware operators.

By splitting a set of mappings $\Omega^P_\emptyset$ into two joined sets of mappings $\Omega^{P'}_\emptyset \bowtie \Omega^{P''}_\emptyset$, where P' is the graph pattern containing all the ranked triple patterns and P'' is the pattern containing the rest, we can apply the algebraic law that pushes ρ over $\bowtie$. Applying this law to a ROB plan of the form:

$$\rho_{b_1}(\rho_{b_2}(\ldots(\rho_{b_n}(\Omega^{P'}_\emptyset \bowtie \Omega^{P''}_\emptyset))))$$

we obtain a plan of the form:

$$\rho_{b_1}(\rho_{b_2}(\ldots(\rho_{b_n}(\Omega^{P'}_\emptyset)))) \bowtie \Omega^{P''}_\emptyset$$

where the $\bowtie$ operator needs to preserve only the order of the first operand. Thus it is possible to use a streaming index join between the two sets of mappings.

The splitting strategy can be redefined to adopt the standard planning heuristic of avoiding Cartesian joins: if there is no shared variable between two ranked triple patterns, the strategy must include into the ranked BGP also the "bridge" triple patterns that have a shared variable with each of them (or chains of triple patterns). Fig. 4.c provides an example of application of the `INTER` strategy on the running case. Since the ranked triple patterns *(?pr hasA1 ?a1)* and *(?of hasA1 ?p1)* have no shared variable, we include *(?of hasP1 ?pr)* as a bridge triple pattern inside the ranked sub-plan. The triple pattern *(?pr hasN ?n)* does not influence the final order, so it can be safely put in the non-ranked sub-plan.

Rank-Join Strategy. The Rank-join strategy (`RJ`) involves the use of one or more rank-join operators in a query plan. This strategy is a variant of the `INTER` strategy, in which the n ranked triple patterns of the query graph pattern P are split each into one BGP P^i (plus the "bridge triple patterns" to avoid Cartesian joins) for $i = 1..n$ and there is a graph pattern P^{n+1} containing the rest of P.

Following this rule, `RJ` first splits a given set of mappings $\Omega_\emptyset^P$ into $n+1$ joined sets of mappings $\Omega_\emptyset^{P^1} \bowtie \cdots \bowtie \Omega_\emptyset^{P^{n+1}}$, where n is the number of ranking criteria. Then, `RJ` can apply the law pushing ρ over $\bowtie$ to obtain a plan of the form:

$$(((\rho_{b_1}(\Omega_\emptyset^{P^1}) \bowtie \rho_{b_2}(\Omega_\emptyset^{P^2})) \bowtie \cdots \bowtie \rho_{b_n}(\Omega_\emptyset^{P^n})) \bowtie \Omega_\emptyset^{P^{n+1}})$$

where all the $\bowtie$ operators, except the rightmost, are rank-joins as they operate on ranked sub-plans; the rightmost $\bowtie$ operator needs to preserve only the order of the first operand, thus an index join can be used. If the datastore already provides a *sorted access* according to b_i to $\Omega_\emptyset^{P^i}$, then the retrieved set of mappings is already a ranked set of mappings $\Omega_{b_i}^P$, and the planner can remove the corresponding ρ_{b_i} operator from the plan. Fig. 1.b shows the plan generated by `RJ` on the query in Listing 1.1.

`RJ` builds a left-linear tree of operators and selects each rank-join operator using a simple heuristic based on the availability of sorted or random access to ranked sets of mappings. This heuristic assumes that using more random access leads to better performances; the assumption holds if random access has a low cost (as in native triple stores), because random access allows tightening the upper-bound of the score. `RJ` applies: 1) RA-HRJN, when the left input is a ranked mapping $\Omega_{b_i}^P$, in which b_i is a single ranking criterion and P is a triple pattern or a BGP; 2) RSEQ, when the left input is Ω_B^P, in which B is a set of ranking criteria with $|B| \geq 2$, e.g. if the input is another rank-join operator, or if $|B| = 1$ and P is not a a triple pattern or BGP. A more complex strategy could be devised by taking into account also the cost of the sorted access on the right input, but this is left to future work.

The `RJ` strategy also exploits an optimization of the random access on a BGP as, at run-time, it creates a reordered version of the original BGP as follows: first, the triple pattern containing the joining attribute is evaluated; then a random access on the other triples is performed iteratively.

5 Evaluation

This section presents an evaluation of ARQ-$\mathcal{R}$ANK, a prototype implementation of a $\mathcal{S}$PARQL-$\mathcal{R}$ANK query engine that extends Jena ARQ 2.8.9. The ARQ-$\mathcal{R}$ANK source code, the test datasets and the queries used in the experiments are available for download at `http://sparqlrank.search-computing.org/`.

Our experiments are based on a modified version of the Berlin SPARQL Benchmark (BSBM [3]). The BSBM dataset generator has been modified to create additional attributes, generated as aggregates from the existing data (e.g. the average of the ratings for a given product) or according to different probability distributions. We report on the performance of ARQ-$\mathcal{R}$ANK using the *SumDepth* and *wall clock execution time* metrics. The SumDepth metric provides a system-independent measure of the I/O cost of a query, as it sums the total number of processed mappings. The experiments were conducted on an AMD 64bit processor with 2.66 GHz and 4 GB RAM, a Debian distribution with kernel 2.6.26-2, and Sun Java 1.6.0 with 2 GB maximum heap size for the JVM. The reported execution times are the average of 20 tests, measured after 5 warm ups, removing the outlier values according to the three sigma rule. The experiments were executed using Jena TDB 0.8.11 as a native triple storage.

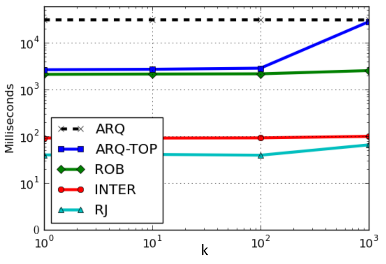
5.1 Rank-Aware Optimization Strategies Evaluation

In the following we evaluate the performance of the `ROB`, `INTER` and `RJ` strategies w.r.t. to the non-optimized Jena ARQ 2.8.8 (`ARQ`) and the TopK optimization [5] introduced in Jena ARQ 2.8.9 (`ARQ-TOP`) on a five million triples dataset.

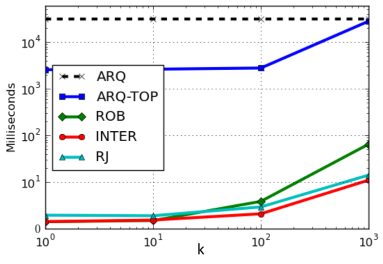
Fig. 5 reports the results of experiments performed on the query in Listing 1.1, where ranking criteria follow a *normal* ($\mu = 0.5, \sigma = 0.15$) score distribution, at varying values of k (we consider also $k = 1000$, while in real world applications k is typically less than 100). The reported numbers represent the complete evaluation time of the query, without considering the planning time.

Fig. 5.a depicts the results obtained when no sorted access indexes are available in the data storage. This setting represents the worst-case scenario for the usage of ARQ-$\mathcal{R}$ANK, as rank (e.g. MPro) and rank-join (e.g. HRJN) algorithms are proven to be efficient when sorted access is available for at least one operand. Despite the additional overhead required to provide an initial ordering to the mappings fetched from the data storage, all the techniques show a performance increase (from 0.2x to 10x), a result that is mainly accountable to the reduced number of mappings that flow in the query engine, as shown in Fig. 6.

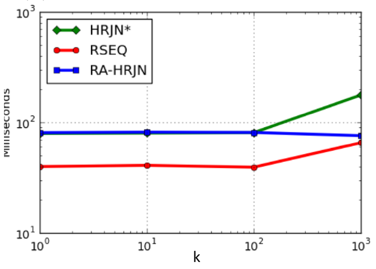
In Fig. 5.b we report the results of the query when sorted access indexes are available. In this configuration, the number of mappings extracted from the data storage significantly decreases (as shown in Fig. 6 SumDepth ranges from 10-1000 in `RJ`, as opposed to 10^5 in `ARQ-TOP`) as results are already ordered according to one of the ranking criteria, thus exploiting at full potential the features of the adopted rank and rank-join operators: all the proposed strategies consistently outperform `ARQ-TOP` up to three orders of magnitude.



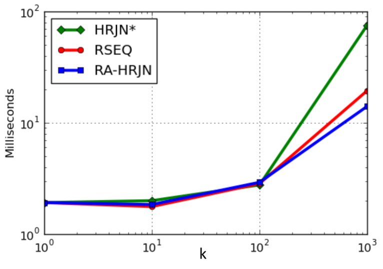
(a) ExTime without sorted access indexes.



(b) ExTime with sorted access indexes.



(c) ExTime of rank-join algorithms without sorted access indexes.



(d) ExTime of rank-join algorithms with sorted access indexes.

Fig. 5. Performance evaluation for the optimization techniques of Section 4.3.

In Fig. 5.c-d we report the results of a comparison analysis for the rank-join operators HRJN*, RA-HRJN, and the proposed RSEQ algorithm. The natural availability of efficient random access to the underlying triple store provides great benefits in terms of accessed mappings and required execution time. Note that the RSEQ and RA-HRJN operators get more effective as k increases. When no sorted access indexes are available, as shown in Fig. 5.c, RSEQ is faster than RA-HRJN, because it requires sorted access only on the left input, while RA-HRJN needs sorted access on both inputs. Fig. 5.d presents the case in which sorted access indexes are available. In this case, RSEQ is comparable to RA-HRJN.

Although the selected case study query may account for the reported numbers, the overall (and consistent) performance increase w.r.t. `ARQ-TOP` provides strong evidence about the general applicability of the proposed approach, also when no sorted access indexes are available. The selection of the best configurations depends on several factors, such as the costs of random and sorted access (which differ based on the storage system), distribution of scores, and others. A detailed study on SPARQL top-k query planning is left to future work.

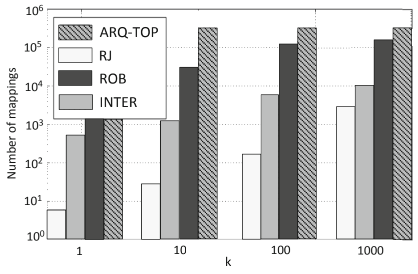


Fig. 6. SumDepth with sorted access indexes

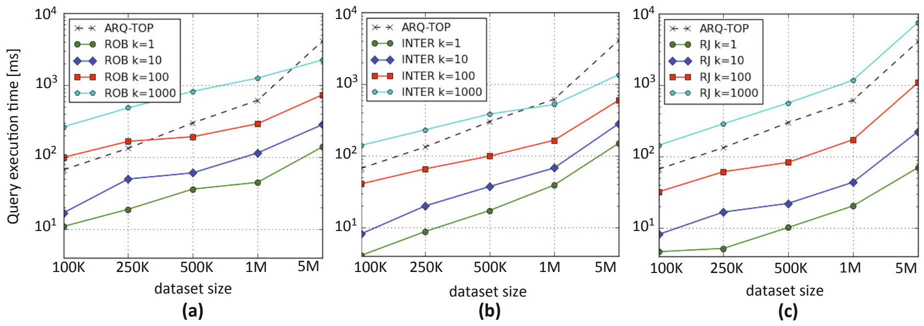


Fig. 7. Performance Evaluation of ARQ-$\mathcal{R}$ANK on the benchmark query mix using the a) ROB, b) INTER, and c) RJ optimization strategies

5.2 Query Benchmark Evaluation

As far as we know, currently there is no benchmark for top-k queries SPARQL. Therefore, we created a small benchmark of 8 queries[7] having 2 and 3 ranking criteria. Although previous works [18] show that the cost associated with top-k algorithms is mainly related to the selected k value, to provide asymptotic evidence of the utility of ARQ-$\mathcal{R}$ANK, we performed the evaluation on five datasets of increasing size (from 100K triples to 5M triples). Fig. 7 depicts the average execution times (for each considered dataset) of the query mix in ARQ-TOP and using the ROB, INTER, and the RJ strategies.

When $1 \leq k \leq 100$, typical values in top-k queries, ARQ-$\mathcal{R}$ANK consistently outperforms ARQ-TOP (up to two orders of magnitude), with a gain proportional to the dataset size. The RJ optimization strategy yields considerably better performance for $1 \leq k \leq 10$, while the INTER strategy is the best for $k = 100$. Note that the ROB strategy provides significant performance gain, while requiring a negligible planning effort.

On the other hand, when $k = 1000$ and the dataset size is below 1M triples, the two systems show comparable performance; nonetheless, the ROB and INTER strategies show a performance gain w.r.t. ARQ-TOP for the biggest dataset in the experiment, and the curves suggest an improvement also for bigger datasets.

6 Conclusion and Future Work

In this paper, we addressed the problem of efficiently executing top-k SPARQL queries; we introduced an incremental execution model for the $\mathcal{S}$PARQL-$\mathcal{R}$ANK algebra and we presented ARQ-$\mathcal{R}$ANK, a rank-aware SPARQL query engine that builds on the $\mathcal{S}$PARQL-$\mathcal{R}$ANK algebra and exploits state-of-the-art rank-aware query operators. We proposed RSEQ, a rank-join algorithm specifically designed to exploit fast random access in native triple stores, and we analyzed the behavior of the system under several configurations. Our results show that ARQ-$\mathcal{R}$ANK is

[7] Queries are available at http://sparqlrank.search-computing.org/

consistently able to significantly improve the performance of top-k queries. Although our evaluation did not include other SPARQL engine implementations, the proposed execution model and planning strategies are applicable to other pull-based SPARQL query engine implementations.

The results presented in this paper are part of a broader research work, where we plan to study more advanced, cost-based, optimization techniques that estimate the cardinality of intermediate results in multiple pipelined rank-join operator evaluations. Among our goals there is also a study on efficient materialization of RDF views for sorted access, and the effects of the relaxation of the *certain variables* constraint defined in Section 3, as the introduction of potentially unbound variables brings uncertainty in the score evaluation. Finally, we outlook extensions of $\mathcal{S}$PARQL-$\mathcal{R}$ANK w.r.t. SPARQL 1.1 federation extension.

Acknowledgements. This research is supported by the Search Computing project, funded by ERC, under the IDEAS Advanced Grants program.

References

1. Bozzon, A., et al.: Towards and efficient SPARQL top-k query execution in virtual RDF stores. In: DBRANK Workshop in VLDB 2011 (2011)
2. 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)
3. Bizer, C., Schultz, A.: The Berlin SPARQL Benchmark. Int. J. Semantic Web Inf. Syst. 5(2) (2009)
4. Li, C., et al.: RankSQL: query algebra and optimization for relational top-k queries. In: SIGMOD 2005. ACM (2005)
5. Castagna, P.: Avoid a total sort for order by + limit queries. JENA bug tracker, `https://issues.apache.org/jira/browse/jena-89`
6. Della Valle, E., et al.: Order matters! harnessing a world of orderings for reasoning over massive data. Semantic Web Journal (2012)
7. Hwang, S.-W., Chang, K.: Probe minimization by schedule optimization: Supporting top-k queries with expensive predicates. IEEE TKDE 19(5) (2007)
8. Ilyas, I.F., et al.: Rank-aware Query Optimization. In: SIGMOD 2004. ACM (2004)
9. Ilyas, I.F., et al.: A survey of top-k query processing techniques in relational database systems. ACM Comput. Surv. 40(4) (2008)
10. Cheng, J., Ma, Z.M., Yan, L.: f-SPARQL: A Flexible Extension of SPARQL. In: Bringas, P.G., Hameurlain, A., Quirchmayr, G. (eds.) DEXA 2010, Part I. LNCS, vol. 6261, pp. 487–494. Springer, Heidelberg (2010)
11. Pérez, J., et al.: Semantics and complexity of SPARQL. ACM Trans. Database Syst. 34(3) (2009)
12. Anyanwu, K., et al.: SemRank: ranking complex relationship search results on the semantic web. In: WWW 2005. ACM (2005)
13. Schmidt, M., et al.: Foundations of SPARQL query optimization. In: ICDT 2010. ACM (2010)
14. Stocker, M., et al.: SPARQL basic graph pattern optimization using selectivity estimation. In: WWW 2008. ACM (2008)
15. Martinenghi, D., Tagliasacchi, M.: Cost-Aware Rank Join with Random and Sorted Access. IEEE TKDE (2011)

16. Bruno, N., et al.: Evaluating Top-k Queries over Web-Accessible Databases. In: ICDE 2002. IEEE (2002)
17. Lopes, N., Polleres, A., Straccia, U., Zimmermann, A.: AnQL: SPARQLing Up Annotated RDFS. 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. 518–533. Springer, Heidelberg (2010)
18. Schnaitter, K., Polyzotis, N.: Optimal algorithms for evaluating rank joins in database systems. ACM Transactions on Database Systems 35(1) (2010)
19. Straccia, U.: SoftFacts: A top-k retrieval engine for ontology mediated access to relational databases. In: SMC 2010. IEEE (2010)
20. Siberski, W., Pan, J.Z., Thaden, U.: Querying the Semantic Web with Preferences. 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. 612–624. Springer, Heidelberg (2006)
21. Qi, Y., et al.: Sum-Max Monotonic Ranked Joins for Evaluating Top-K Twig Queries on Weighted Data Graphs. In: VLDB 2007 (2007)

Collaborative Filtering by Analyzing Dynamic User Interests Modeled by Taxonomy

Makoto Nakatsuji[1], Yasuhiro Fujiwara[2], Toshio Uchiyama[1], and Hiroyuki Toda[1]

[1] NTT Service Evolution Laboratories, NTT Corporation
[2] NTT Software Innovation Center, NTT Corporation, 1-1 Hikari-no-oka, Yokosuka-Shi, Kanagawa, 239-0847 Japan
{nakatsuji.makoto,fujiwara.yasuhiro,uchiyama.toshio, toda.hiroyuki}@lab.ntt.co.jp

Abstract. Tracking user interests over time is important for making accurate recommendations. However, the widely-used time-decay-based approach worsens the sparsity problem because it deemphasizes old item transactions. We introduce two ideas to solve the sparsity problem. First, we divide the users' transactions into epochs i.e. time periods, and identify epochs that are dominated by interests similar to the current interests of the active user. Thus, it can eliminate dissimilar transactions while making use of similar transactions that exist in prior epochs. Second, we use a taxonomy of items to model user item transactions in each epoch. This well captures the interests of users in each epoch even if there are few transactions. It suits the situations in which the items transacted by users dynamically change over time; the semantics behind classes do not change so often while individual items often appear and disappear. Fortunately, many taxonomies are now available on the web because of the spread of the Linked Open Data vision. We can now use those to understand dynamic user interests semantically. We evaluate our method using a dataset, a music listening history, extracted from users' tweets and one containing a restaurant visit history gathered from a gourmet guide site. The results show that our method predicts user interests much more accurately than the previous time-decay-based method.

1 Introduction

User interests can switch rapidly. For example, some one who listens to "avant garde" music may switch to "relax" music according to his/her mood at that time. The user's transaction history may thus contain several dissimilar strings, each of which is identified by the similarity of contiguous item selections. This creates a significant problem in collaborative filtering.

Accurately identifying and handling changes in user interests over time is an active research challenge in recommender systems, and is the target of many studies [6,8,14,15,22,24,29,31]. One major research approach is to use a time decay function that decreases the item weight with the item's age [6,14,15]. Time decay methods assume that the recent item transactions of the active user, the

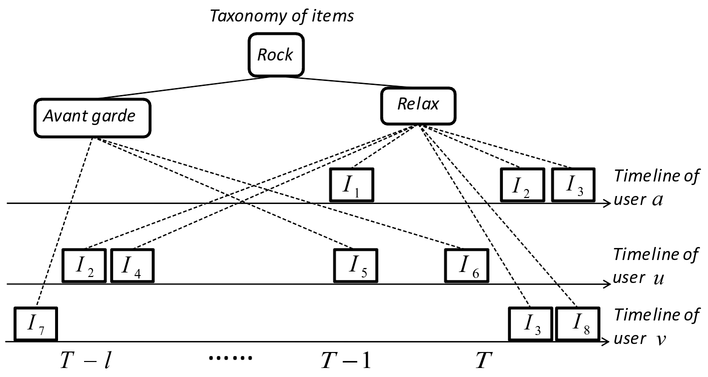


Fig. 1. Transaction timelines of users. T is the current epoch. Boxes indicate artists (items) and the subscript identifies the artist; the dotted lines indicate music classes according to the music taxonomy. For example, in the current epoch, user a listened to songs of artist I_2 and artist I_3 (both in class "Relax").

one who is to receive the recommendation, reflect his/her future interests more than old transactions. Thus, they gradually decay the influence of old data.

Previous time-decay-based methods, however, worsen the sparsity problem in collaborative filtering, which is well known to produce low recommendation accuracy when the population of the dataset used to measure the similarity of users is not sufficient [27]. The transaction timing of items is usually different for each user and for each item as described in the diffusion of innovations [26]. We consider that the sparsity problem occurs due to such *time offsets* against the item transactions of users in the real world. We illustrate the problem using Figure 1. Previous time-decay-based methods compute the interests of user v as being similar with those of active user a because they transact the same item, I_3, in the same current epoch T. Previous methods also indicate that the interests of user u are dissimilar to those of user a because they share no items in recent epochs. This implies that users are seen as similar only if they transact the same items in recent epochs. However, there are few users who have transacted the same items with user a in recent epochs (like user v), thus the time-decay-based methods suffer badly from the sparsity problem.

This paper proposes a novel method that overcomes the sparsity problem; it avoids the problems that occur when using temporal information of item transactions to improve recommendation accuracy. Our method has two ideas. First, it extracts, in a per epoch manner, transactions that are similar to the transactions of the current epoch of the active user from all transactions of other users regardless of age. This has the effect of eliminating dissimilar transactions in epochs while it can make use of similar transactions regardless of age. For example, in Figure 1, it can eliminate the transactions made by user u in epoch T while it can make use of those made by u in epoch $T-l$ because they transact the same item, I_2. Second, it models transactions of items based on a taxonomy of items, which is sometimes called the "simple ontology" [17]; it is a collection of human-defined classes usually with hierarchical structure. Our method makes use of the class structure in the taxonomy and thus can measure the similarity of transactions in epochs from both transacted items and their classes. As described in [18], this avoids the sparsity problem since more data is available for similarity matching. Moreover, it suits the computation of recommendations under the situation that the items transacted by users dynamically change over time. This is because the semantic meaning behind classes does not change so often while items transacted by users dynamically change over time. For example, active user a recently transacted item I_2 that was selected by user u who also selected item I_4 in the same epoch as I_2 (that is, epoch T–l), and I_4 shares the class of "Relax" with I_2. Thus, our method measures transactions by user a in epoch T and those by user u in epoch $T-l$ as similar because those transactions are dominated by class "Relax" in addition to the fact that those transactions include item I_2. The semantic meaning behind class "Relax" does not change over time, so our method can identify I_4, which is missing in epoch T, as of potential interest to user a. Those two ideas enable our method to overcome the sparsity problem while suppressing noisy transactions, and thus achieve high accuracy.

Taxonomies provide another attractive effect. They enrich semantics behind the recommendations. The users can understand the recommended items as belonging to the same classes as the items that the user has transacted recently. Those taxonomies are becoming available on the Web due to the spread of the Linked Open Data (LOD) vision [1]. For example, Freebase[1] and DBPedia [2] have detailed taxonomies against several domains such as music, movie, and books. As an example, music genre "Electronic dance music" in FreeBase is identified by the unique resource identifier (URI)[2] and is available in RDF format. By referring to this URI, the computer can acquire the information that "electronic_dance_music" has "electronic_music" as parent_genre, "house_music" as one of its subgenres, and "pizzicato_five" as one of its artists as well as having the owl:sameAs relationship with "Electronic_Dance_Music" in DBPedia. Why don't we use those taxonomies for semantically analyzing dynamic user interests in the era of the "Web of Data"?

To the best of our knowledge, this is the first study that employs a taxonomy of items and uses discrete time periods to isolate changes in interest over time. We consider that analyzing dynamic user interests over taxonomies (or simple ontologies) is very important since taxonomy is a core Semantic Web technology. Our idea is simple but provides accurate recommendations. It provides a new theoretical alternative to collaborative filtering techniques that use temporal information in making recommendations.

We applied our ideas to the widely used service of neighborhood-based collaborative filtering. We evaluated our method using the following two datasets: (1) a dataset of music listening history extracted from users' tweets at twitter[3]

[1] `http://www.freebase.com`

[2] `http://rdf.freebase.com/rdf/en/electronic_dance_music`

[3] `http://twitter.com/`

with a taxonomy created from last.fm[4] tags[5] and (2) one containing restaurant visit histories gleaned from a popular Japanese gourmet guide site, Tabelog[6] with an expert created taxonomy. Those taxonomies have the same structures as the LOD dataset in Freebase as explained above. The results show that our method outperforms previous time-decay-based methods. They also indicate that our taxonomy-based method is superior to the typical topic model, LDA (Latent Dirichlet Allocation) method [3], which estimates user interests from data-driven topics, and is also successful in overcoming the sparsity problem of collaborative filtering. This is because topics estimated from users' item transactions dramatically change over time whereas the semantics behind human-defined classes do not change so often.

The paper is organized as follows: we describe related works in the next section. Section 3 describes the background of this paper. Section 4 explains our method in detail; how to measure the similarity of current transactions of the active user and the epoch-based transactions of another user by using a taxonomy of items. Section 5 evaluates our method in detail. Finally, Section 6 concludes the paper.

2 Related Work

Recently, several works have attempted to integrate temporal dynamics into collaborative filtering methods [6,8,14,15,22,24,29,31]. One of the major research approaches in this field uses a time decay function and computes the time weights for different items by decreasing the weights according to data age [6,14,15]. Recently, Liu et al. proposed an incremental algorithm for updating neighborhood similarities given new data [15]. While our method is not a time-decay-based method, it can be combined with those to catch the trends in item transactions. To this end, it is necessary to apply time decay functions to item transactions according to their transacted epochs, after extracting transactions similar to those in the current epoch of the active user.

Other research studies use state-based models. Markov chain models have been widely applied to the next-page prediction problem [24,31]. Topic Tracking Model [8], which extends LDA [3], uses state space models on the natural parameters of the multinominal distributions that represent the topics. Recently, [24] introduced a novel personalized Markov chain method that models a transition cube, where each slice is a user-specific transition matrix of an underlying Markov chain on the users' basket history. They introduce a factorization model that gives a low-rank approximation of the transition cube. The quality of the final transition graph, and thus the accuracy of item prediction, is much improved since the influence of transitions of similar users, similar items, and similar transitions is considered. However, those factorization models can not provide semantics under the recommendation results such as the semantic relationships between recommended items and items that the user has transacted

[4] http://www.last.fm/
[5] The music taxonomy can be acquired, only for academic use, by mailing the authors.
[6] http://tabelog.com/

recently. [14] proposed a method that applies a time-decay method to a factorization model with complicated parameter learning from explicit rating datasets. It learns parameters for the biases of each user and each item, both of which change over time. However, generalizing factorization models to handle implicit feedback data only achieves a slight improvement [7]. Thus, it is not well suited for the implicit ratings used in our evaluation.

There are several alternatives that use temporal information in making recommendations. [22] proposed a preceding mining model that exploits the mined precedence information to recommend the top-k choices that could follow the past choices of a particular user. It models the user's history as a set of items that occurred in the past instead of a strict sequence of items, and predicts the set of items most likely to follow in no particular order. [29] proposed a graph model that captures users' long-term and short-term interests over time. It balances the impacts of long-term and short-term interests for accurate recommendations.

We note that taxonomies are becoming available on the Web in the format of LOD such as those published by DBPedia [2] and Freebase. Though most of the data published in the format of LOD is instance data, there are projects that link and build taxonomies by merging the data in several domains by using already published taxonomies like those in Wikipedia[7] [9,23]. Such merged taxonomies enable us analyze user transactions distributed in multiple service domains comprehensively. Thus, recommendation methods that use taxonomies to understand user interests semantically are becoming more important [4,18–21,28,30]. For example, [18] measures the similarity of users based on items rated by users as well as the classes that include those items. Thus, it accurately predicts user interests even when the rating dataset is sparse. They recently proposed a method that analyzes user interests more in detail by linking multiple taxonomies [19]. However, they fail to handle temporal information against item transactions.

This paper differs from the methods that apply a time-decay function against users' item transactions, methods that use state-based models, and techniques that focus on the sequence of item transactions. Our method employs the taxonomy of items and extracts transactions that are similar to the transactions of the current epoch of the active user. Our ideas give a new vision of collaborative filtering by better utilizing the temporal information of item transactions.

3 Background

Collaborative filtering methods can be classified into two approaches: memory-based (or neighborhood-based) collaborative filtering [6,15,25] and model-based collaborative filtering [8, 24, 31]. Previous time-weighted collaborative filtering methods [6,15] are examples of memory-based collaborative filtering. Our method, however, can also be applied to model-based collaborative filtering[8].

[7] `http://en.wikipedia.org`

[8] For example, we can create an order-three tensor from transactions per epoch by users, items, and their classes. By applying tensor decomposition [12], we can compute recommendations against the current transactions of the active user.

Table 1. Definition of main symbols

Symbols	Definitions
t	An epoch
T	A current epoch
$\mathcal{C}$	A class set in the taxonomy
I_j	An item
C_j	A class in the class set $\mathcal{C}$
$\mathbf{i_u^t}$	A vector of item transactions of user u in epoch t
$\mathbf{c_u^t}$	A vector of class transactions of user u in epoch t

Traditional memory-based collaborative filtering methods assume that each user belongs to a larger group of users with similar behavior [25]. In computing user similarity, they often use cosine similarity.

If $\mathcal{I}$ is the set of items transacted by users a and u, and $i_{u,j}$ is the transaction frequency of user u for item I_j, the similarity between active user a and user u, $S(a,u)$, is determined as follows:

$$S(a,u) = \frac{\sum_{I_j \in \mathcal{I}}(i_{a,j} \cdot i_{u,j})}{\sqrt{\sum_{I_j \in \mathcal{I}}(i_{a,j}^2)}\sqrt{\sum_{j \in \mathcal{I}}(i_{u,j}^2)}}. \tag{1}$$

If $\mathcal{N}$ is the set of users that are most similar to user u, the predicted value of user a on item I_j, $p_{a,j}$, is given by the following equation:

$$p_{a,j} = \frac{\sum_{u \in \mathcal{N}} (i_{u,j} \cdot S(a,u))}{\sum_{u \in \mathcal{N}} S(a,u)}. \tag{2}$$

This equation implies that the methods recommend items based on user similarities. Therefore, the effective assessment of user similarities is important in improving recommendation accuracy. Our method, explained in the next section, extracts transactions that are similar to the transactions of the current epoch of the active user. Thus it can produce more accurate recommendations than the ordinary cosine based method as shown in the evaluation section.

4 Method

We first explain our model of item transactions in an epoch according to a taxonomy of items. Next, we show how to extract transactions that are similar to the transactions in the current epoch of the active user. We then introduce recommendation computation by analyzing extracted transactions.

4.1 Modeling Transactions of a User in an Epoch

We explain how to model transactions by a user in an epoch. Please refer to the symbol definitions of Table 1.

We assume that epoch t is a discrete variable, a time period, and we can set the time period for an epoch arbitrarily at, for example, one day or one week as [8] did. An epoch can overlap adjacent epochs by offsetting the starting time of epochs. This is useful because we try to analyze the change in user interests in detail. We also denote T as the current epoch.

Our model is based on two observations. First is that a user who transacts an item within an epoch, tends to like items of the same class in that epoch. In other words, users tend to be interested in the same types of items in the same period of time. For example, users who are interested in Opera music item in a certain epoch, tend to transact several Opera items in the same epoch. Second is that the semantics behind the classes do not change so often while the items transacted by users appear and disappear in each epoch as influenced by the trends in the epoch. For example, the class "Metal Rock" is used with almost the same meaning for a long period while particular artists in "Metal Rock" often appear and disappear. Users who like "Metal Rock" tend to like items that were classified into "Metal Rock" even if those items were transacted by other users in different epochs. Taxonomy-based modeling is thus useful in analyzing temporal user interests because it lets our method measure the similarity of transactions in different epochs by using classes. Thus, we propose to use a taxonomy of items to model a user's temporal interests from his/her item transactions in epochs. As a result, our method can identify similar transactions in epochs by using both items and their classes as described in Section 4.2. Thus it can overcome the sparsity problem that occurs when there are few item transactions in each epoch even if the items transacted by users dynamically change over time.

Formally, our method models the temporal interests of a user by using two vectors; a vector of item transactions of user u in epoch t, $\mathbf{i_u^t}$, and a vector of class transactions of user u in epoch t, $\mathbf{c_u^t}$. The j-th element of vector $\mathbf{i_u^t}$, $i_{u,j}^t$, represents the frequency of transactions of item I_j in epoch t. The k-th element of vector $\mathbf{c_u^t}$, $c_{u,k}^t$, represents the frequency of transactions of class C_k in epoch t. $c_{u,k}^t$ is computed by the following equation:

$$c_{u,k}^t = \sum_{I_j \in f(k)} i_{u,j}^t. \tag{3}$$

Here, function $f(k)$ returns an item set whose members belong to descendant classes of class C_k and have unique names (no names are shared)[9]. Equation (3) reflects the transaction frequencies of items in their ascendant classes. Thus, we can measure the similarity of transactions of users in epochs from classes as well as from items, which overcomes the sparsity problem.

4.2 Measuring Similarities of Transactions in Epochs

We explain here how to compute the similarity between the transactions in current epoch T of active user a and those in epoch t of user u. To avoid the

[9] Some items in the taxonomy may have the same name, however, they should be identified by URIs. For example, some artists may be placed into several classes, however, each artist in a different class should be identified by a URI, which is assigned to that artist regardless of its name.

Algorithm 1. Measuring transaction similarity in the current epoch of user a and transactions in all epochs of user u.

```
Input: Transaction vectors of user $a$ in epoch $T$, ($\mathbf{i_a^T}$ and $\mathbf{c_a^T}$) and transaction vectors in epoch $t$ of
    user $u$, ($\mathbf{i_u^t}$: $0 \leq t \leq T$) and ($\mathbf{c_u^t}$: $0 \leq t \leq T$)).
Output: Similarity score between transactions in the current epoch of user $a$ and those in epoch $t$
    of user $u$.
1: for each epoch ($t : 0 \leq t \leq T$) do
2:   for each user $u$ do
3:     for each class $C_j$ in $\mathcal{C}$ do
4:       for each sub-class $C_k$ in $\{\mathcal{C}_{j,T}(a) \cup \mathcal{C}_{j,t}(u)\}$ do
5:         compute $S(a, u, C_k, t)$;
6:       end for
7:       for each item $I_k$ in $\{\mathcal{I}_{j,T}(a) \cup \mathcal{I}_{j,t}(u)\}$ do
8:         compute $S(a, u, I_k, t)$;
9:       end for
10:    end for
11:    Compute similarity score $S_C(a, u, t)$;
12:    Compute similarity score $S_I(a, u, t)$;
13:  end for
14:  for each user $u$ do
15:    Normalize $S_C(a, u, t)$ and $S_I(a, u, t)$;
16:    Compute $S(a, u, t)$ as $S'_C(a, u, t) + S'_I(a, u, t)$;
17:  end for
18: end for
```

sparsity problem, we measure the similarity of transactions in epochs using both transacted items and their classes. We also apply set theory [11] to analyze the class structure, which is composed by class/sub-class relationships and class/item relationships in the taxonomy, to assess user similarities in detail. We first give the notations for the algorithm and then explain it in detail.

Notation. Our algorithm applies set theory to assess the similarity of users' interests in epochs according to the class structure of the taxonomy, which is composed by class/sub-class and class/item relationships in the taxonomy. Thus, we introduce notations to represent those relationships. We denote $\mathcal{C}_{j,t}(u)$ as a sub-class set whose members belong to class C_j and that have been transacted by user u in epoch t. Thus, $\{\mathcal{C}_{j,T}(a) \cup \mathcal{C}_{j,t}(u)\}$ and $\{\mathcal{C}_{j,T}(a) \cap \mathcal{C}_{j,t}(u)\}$ are a union set and an intersection set of sub-classes of class C_j transacted by user a in epoch T and those by user u in epoch t, respectively. We also denote $\mathcal{I}_{j,t}(u)$ as an item set whose items belong to class C_j and that have been transacted by user u in epoch t. Thus, $\{\mathcal{I}_{j,T}(a) \cup \mathcal{I}_{j,t}(u)\}$ and $\{\mathcal{I}_{j,T}(a) \cap \mathcal{I}_{j,t}(u)\}$ are a union set and an intersection set of items in class C_j transacted by user a in epoch T and those by u in epoch t, respectively.

Algorithm. Our method measures the similarity of interests in epochs using both transacted items and their classes. Please also see Algorithm 1. The algorithm proceeds in the following steps:

1. Our method measures the similarity between the transactions in the current epoch of the active user and those in each epoch of the other users. Thus, it repeats steps from 2 to 6 by changing time t from 0 to T (line 1) and by setting target user u as all users (line 2).
2. It also picks up class C_j among the class set $\mathcal{C}$ in the taxonomy (line 3).
3. For each sub-class C_k in class C_j, our method computes the similarity between interests of user a in current epoch T and those of user u in epoch t. We assume that if users transact the same amount of transactions against a class in an epoch, they have similar interests to the class. Thus, the similarity of users' interests against sub-class C_k in epochs is not high if the transactions of user u have too many or too few transactions against sub-class C_k in epoch t compared with transactions against C_k by user a in epoch T. Thus, we design this similarity, denoted as $S(a, u, C_k, t)$, to filter such transaction noise and take the smaller of the class transaction frequencies between users in epochs as follows (line 4-6):
$$S(a, u, C_k, t) = min(c^T_{a,k}, c^t_{u,k}). \tag{4}$$
4. As is the case with $S(a, u, C_k, t)$, for each item I_k, our method computes the similarity between the interests in epoch T of user a and the interests in epoch t of user u. This similarity, denoted as $S(a, u, I_k, t)$, is formally computed as follows (line 7-9):
$$S(a, u, I_k, t) = min(i^T_{a,k}, i^t_{u,k}). \tag{5}$$
5. Next, it computes the similarity between the class transactions in the current epoch T of active user a and the class transactions in epoch t of user u. This similarity, $S_C(a, u, t)$, is computed as follows (line 11):
$$S_C(a,u,t) = \sum_{C_j \in \mathcal{C}} \frac{\sum_{C_k \in \{\mathcal{C}_{j,T}(a) \cap \mathcal{C}_{j,t}(u)\}} S(a, u, C_k, t)}{|\{\mathcal{C}_{j,T}(a) \cup \mathcal{C}_{j,t}(u)\}|}. \tag{6}$$
The numerator sums the similarity of users' interests in epochs against each sub-class C_k in class C_j computed in step 3. The denominator, which represents the number of members of a set $\{\mathcal{C}_{j,T}(a) \cup \mathcal{C}_{j,t}(u)\}$, lets our method measure the similarity of users in epochs against class C_j considering the overlap between the sub-classes of a class C_j transacted by user a and those by user u. By picking up each class C_j in $\mathcal{C}$ and by investigating the similarity of users' interests in each epoch against each class/sub-class relationship, we can make use of the structure of the class hierarchy for measuring the similarity. We consider that the denominator should be added when the taxonomy of items is not so detailed, that is, each of the members of $\mathcal{C}$ has many sub-classes, like our evaluation dataset of music listening history. This is because our method can compute the similarity of users' interests against a class by investigating the overlap rate of its sub-classes owned by users.
6. As is the case with $S_C(a,u,t)$, it computes the similarity between the current item transactions of active user a and the item transactions in epoch t of user u. This similarity, $S_I(a, u, t)$, is computed as follows (line 12):

$$S_I(a,u,t) = \sum_{C_j \in \mathcal{C}} \frac{\sum_{I_k \in \{\mathcal{I}_{j,T}(a) \cap \mathcal{I}_{j,t}(u)\}} S(a,u,I_k,t)}{|\{\mathcal{I}_{j,T}(a) \cup \mathcal{I}_{j,t}(u)\}|}. \tag{7}$$

We can also use cosine similarity, see Equation (1) in computing the similarity between the current item transactions of active user a and the item transactions in epoch t of user u. In our evaluation, we compared those similarity measurements against item transactions.

7. Our method normalizes the similarity against classes and that against items. Thus, the variance and the average of similarity scores among all epochs of users equal one and zero, respectively. We denote those normalized similarities as $S^{'}_C(a,u,t)$ and $S^{'}_I(a,u,t)$. Next, our method computes the similarity between the transactions in current epoch T of user a and those in epoch t of user u. This similarity, $S(a,u,t)$, is computed as follows (line 14-17):

$$S(a,u,t) = S^{'}_C(a,u,t) + S^{'}_I(a,u,t). \tag{8}$$

Note that our algorithm is fast even though it includes several loops because typically there are few user transactions in each instance or each class in each epoch.

4.3 Computing Recommendation

Our method uses the similarity values computed by Equation (8) to compute a prediction value against item I_j for active user a. The predicted value of user a on item I_j, $p_{a,j}$, is obtained by the following equation:

$$p_{a,j} = \frac{\sum_{u,t \in \mathcal{N}} (i^t_{u,j} \cdot S(a,u,t))}{\sum_{u,t \in \mathcal{N}} S(a,u,t)}, \tag{9}$$

where $\mathcal{N}$ is the set of transactions in epochs of users that are most similar to the current transactions of user a.

This equation is similar to Equation (2) used in traditional memory-based collaborative filtering. However, note that our method computes item prediction from the most similar transactions in epochs not from the most similar users. Similar users, computed by traditional memory-based collaborative filtering, may transact different types of items at different times. Our method can filter out such transaction noise for the active user, and so achieves high accuracy.

5 Evaluation

We conduct an evaluation to confirm the method's accuracy.

5.1 Datasets

Our evaluation used the following two datasets:

Music listening history. We crawled users' tweets against music artists (items) from Twitter from 6th July to 20th September, 2011. To extract users' listening history from tweets, we first extracted artist names from the tweets submitted through last.fm twitter client[10]. Tweets submitted via the client have a fixed format allowing us extract music artist name without error. We also extracted tweets other than those submitted from the client to increase the number of music tweets. For this, we pulled the time-line of tweets of crawled users and checked if those tweets included both music artist name and hashtag "#nowlistening". Users' listening history has been used for evaluating recommendations [13]. By analyzing tweets, we can attach temporal information to the listening history.

The taxonomy of music artists was gathered from last.fm API according to the following procedure: (1) We first crawled the top tags (descending popularity) for each artist. We also crawled 26 genre tags as classes that were located at the top of the "music page" of last.fm to categorize artists. (2) We next crawled the top similar tags for each genre tag and classified those top similar tags as sub-classes of the genre class[11]. (3) We then classified an artist as an instance under sub-classes if their tags were the same as the tags crawled for the artist. For example, if "Beatles" has a tag "Classical rock" and the genre class "Rock" has a similar tag "Classical rock", we create sub-class "Classical rock" under class "Rock" and classify "Beatles" into "Classical rock" as an instance. (4) We resolved the ambiguity caused by tags with the same name. We checked whether an artist had ambiguous tags (i.e. were classified in several genres). If so, we checked the most popular tag for the artist whose name was the same as the genre tag. We also checked whether the ambiguous tag was one of the tags similar to that genre tag. If so, we classified these ambiguous tags into that genre class as sub-classes. The other ambiguous tags were eliminated. For example, if the artist "Beatles" had a tag "classic" but the most popular tag of this artist whose name was the same as the genre tag was "Rock" and "Rock" had "classic" as similar tag, we classified "Beatles" into sub-class "classic" under class "Rock". Finally, we permitted an artist to be classified into not more than three sub-classes in the taxonomy according to the tag popularity for the artist.

As a result, we could extract 62,527 tweets of 14,884 users against 6,886 artists (items) in creating the evaluation dataset. The taxonomy of artists has 1,223 classes and has three hierarchy levels; first hierarchy level is root class. The classes (tags) in the lowest hierarchy in the taxonomy are themselves concrete, however, the parent classes of those classes are not so detailed[12]. Thus, we need the denominator of Equation (6) as explained in the method section.

[10] `http://tweetmlyast.fm/`

[11] Thus, the music taxonomy is really a taxonomy expanded with similar tags created by statistical analysis against tagging activities of last.fm users.

[12] The music taxonomy has 26 genre-classes. Each genre-class has, on average, about 46 sub-classes. We consider that each genre-class can be categorized in more detail.

Restaurant visiting history We also used a restaurant visiting history gathered from the popular Japanese gourmet guide site, Tabelog with its expert-created taxonomy. This dataset was also used for the evaluation of recommendation methods [18,19]. Users submit reviews against restaurants that they liked, along with the date of dining.

We focused on restaurants in Tokyo, and extracted 63,885 reviews of 13,633 users against 44,321 restaurants (items) posted from 9th March to 20th June, 2010. The taxonomy of restaurants is quite deep; it has 318 genres as item classes, and has four or five hierarchy levels. For example, the end classes of this taxonomy have genres such as "Wine bar" and "Beer garden".

5.2 Compared Methods

We compared our method to the following methods:

- *Cosine*: This is the most commonly used memory-based collaborative filtering method; user similarity is based on cosine similarity. The prediction values of items are computed by using Equation (2).
- *Time*: This is the time-weighted collaborative filtering method [6] that uses a time decay function that computes the time weights for different items by decreasing the weights of old data. Time decay is determined by the exponential function $\mathrm{e}^{-(\lambda\cdot(T-t))}$. We selected this method because we consider that our method, which is implemented to realize simple neighborhood-based collaborative filtering should be compared with the simplest and closest methods.
- *Taxonomy*: this taxonomy-based method was proposed by [18]. It computes the similarity of users from the transactions of users against both items and their classes; it does not use time information.
- *LDA*: this method is based on LDA [3], which is a representative topic model, however, it does not use time information. Method *LDA* is model-based collaborative filtering. We selected this method for investigating the characteristics of human-defined classes and those of data-driven topics when applying the methods to the datasets that contain time information.
- *Without classes*: This is the proposed method with the taxonomy of items omitted [13]. This method was chosen for investigating the effect of using taxonomy of items in measuring similarity between transactions in epochs.

5.3 Methodology and Parameter Setup

We divided each dataset into a training dataset and a prediction dataset. The latter contains the data gathered over the last week and former contains the remainder. We then used the training dataset to measure the similarity of transactions in epochs. The starting time of epochs was shifted in units of one week (this means each adjacent epoch has overlap of two weeks) and the length of each epoch was varied from one week to three weeks. The results shown later are

[13] This is equivalent to not summing the first term in Eq. (8) when computing $S(a, u, t)$.

those achieved with the length of epoch set to three weeks because our method achieves the most accurate results at this setting. In concrete, the accuracy degrades if epoch length differs from 3 weeks. Basically, epochs < 3 weeks yield sparse transactions and epochs > 3 weeks yield transactions that include several different classes. We also confirmed that accuracy degrades if the overlap between epochs equals zero because it is difficult to catch the dynamic change of users' interests in detail. We next computed the prediction values of items for the active user by using Equation (9). We focused on users who had both item transactions in the current (last) epoch in the training dataset and those in the prediction dataset as active users. As a result, the music dataset and the restaurant dataset have 1,555 and 2,034 users to be evaluated, respectively.

We used Average Precision (AP) [16] to evaluate our method. If we let the number of ranked items be k, the number of correct answers among the top-j ranked items be N_j, and the number of all correct answers be A (defined as items the user is interested in), AP is defined as $\frac{1}{A}\sum_{1\leq j\leq k}\frac{N_j}{j}$.

We checked AP against the top-k ranked items. We set k to 5, 10, 20, 40, and 60 in our evaluation, and the corresponding results are denoted as AP@5, AP@10, AP@20, AP@40, and AP@60. The active user ordinarily checks the top ranked items in the list, thus typical recommender systems only use those items. We did not check the accuracy against lower ranked items because we consider that the highly ranked items are more important as did a previous work [10]. Some readers may consider that top-40 or top-60 items are too many for the active user. We, however, consider that, in some cases, the recommendation diversity, and hence user satisfaction, is improved by randomly showing about 5 of the 40 or more top ranked items in the recommendation [30].

We set the number of $\mathcal{N}$ (number of most similar *transactions per epoch* used in methods other than *LDA*) in both datasets to 20. We also set parameter λ used in method *Time* to 0.2 for the music dataset and to 0.1 for the restaurant dataset. The number of topics in method *LDA* was set to 20 for both datasets. Those settings maximize the accuracy for each method.

5.4 Results

We evaluated the accuracy of our methods by changing the number of items recommended to the active user. AP@k results against the music dataset are shown in Table 2 and those against the restaurant dataset are shown in Table 3. Our methods (*Proposed* or *Without Classes*) offer better AP@k than the other methods in all cases other than AP@60 of *LDA* method against the restaurant dataset. Bold typeset indicates statistical significance at $p < 0.05$ (t-test was used) compared to *Time* and *LDA*. This indicates that *Proposed* well employs the taxonomy of items and collects the most similar transactions in epochs with the current transactions of the active user while eliminating noisy transactions. We also applied cosine similarity to compute the similarity of item transactions in Equation (5) and (7) in our methods and confirmed that it also has higher accuracy than *Time*. Due to the space limitation, we omit the results of this.

Table 2. Results (x10^{-2}) against music dataset

	AP@5	AP@10	AP@20	AP@40	AP@60
Cosine	1.10	1.35	1.59	1.85	2.00
Time	1.16	1.45	1.63	1.91	2.01
Taxonomy	0.92	1.10	1.52	1.74	1.87
LDA	0.31	0.39	0.48	0.54	0.58
Without classes	*1.18*	*1.52*	*1.67*	1.91	2.01
Proposed	1.14	1.48	1.66	*1.99*	***2.13***

Table 3. Results (x10^{-3}) against restaurant dataset

	AP@5	AP@10	AP@20	AP@40	AP@60
Cosine	1.32	1.44	1.53	1.69	1.77
Time	1.46	1.52	1.63	1.91	2.01
Taxonomy	1.06	1.19	1.27	1.38	1.46
LDA	1.60	1.69	1.82	2.03	*2.14*
Without classes	1.54	1.61	1.74	1.80	1.86
Proposed	*1.63*	***1.78***	***1.90***	*2.08*	2.12

Interestingly, *Proposed* improves the accuracy of higher ranked items more in the restaurant dataset than in the music dataset since the restaurant taxonomy is more detailed than the music one. We also investigated the effect of procedure (4) in Section 5.1 on the recommendation accuracy. This procedure resolves the ambiguity caused by tags with the same name and thus avoids the classification mistakes when expanding the taxonomy with user-generated tags. As a result (we omit the result due to space limitation), we found that the accuracy of the method *Proposed* becomes much worse if we omit procedure (4). Thus, a detailed taxonomy increases the recommendation accuracy of our method.

Our methods also offer higher accuracy than the *Taxonomy* method. This is because *Taxonomy* simply reflects users' transaction frequency against items to their classes and does not consider the time-line of users' transactions. Thus, interests of a user are constructed by the many classes transacted over the entire time-line of the user. This approach is useful when recommending highly novel items that are located in classes that the active user has not yet transacted while maintaining high recommendation accuracy (in terms of Mean Absolute Error) as [18] described. On the other hand, *Proposed* focuses on epochs to improve accuracy. The effect is to suppress the mixing of classes that are not applicable to the current transactions of the active user, which improves accuracy.

Proposed also offers much higher accuracy than the *LDA* method against the music dataset, and it offers higher accuracy than the *LDA* method against the restaurant dataset method except for AP@60. Items transacted by users in the music dataset appear and disappear over short time cycles according to the trends in each epoch. On the other hand, restaurants visited by users do not change so often [14]. From those results, topics estimated by the *LDA* method are

[14] The high accuracy of the *LDA* method for the restaurant dataset has another reason. After investigation, we found that *LDA* could estimate topics based on restaurant location. Users tend to go dining in the same area even in different epochs.

not useful when the item transactions are more dynamic. On the other hand, human-defined classes do not change so often, thus *Proposed* achieves higher accuracy in this situation. Variants of the LDA method have appeared that consider time information [8]. An LDA method that employs the taxonomy of words (items) has been proposed [5]. Thus, we consider that the combination of topics and taxonomies for estimating user interests is promising for future recommendation methods in dynamic situations.

We then investigated the effect of using a taxonomy of items on measuring the similarity between transactions in epochs. To this end, we compared *Proposed* and *Without classes*. For the music dataset, *Without classes* achieves the highest accuracy against AP@5, AP@10, and AP@20 because the music taxonomy is not so detailed, however, it can not achieve higher accuracy against AP@40 and AP@60 than *Proposed* due to the sparsity problem caused by lack of the taxonomy. Another interesting finding is that *Without classes* can not achieve higher accuracy against AP@40 and AP@60 than *Time*. This is because *Without classes* uses only current transactions of the active user to compute recommendations for the active user while *Time* uses all transactions of the active user. Thus, *Without classes* suffers from the sparsity problem. On the other hand, *Proposed* achieves higher accuracy than *Time* in all cases though it also uses only current transactions of the active user. The above results confirm that our method, *Proposed*, avoids the sparsity problem and improves recommendation accuracy by using available taxonomies.

Note that our methods use only item transactions in the current epoch of the active user for comparison purposes. Considering this fact, the above results confirm that our two ideas have great potential for future recommendation techniques that use the temporal information of the users' item transactions; utilizing the transactions of the active user in older epochs can greatly enhance the recommendation accuracy.

Finally, we show examples of recommendations made by only our method, *Proposed*. It can recommend item "Sam Sparro" (in class "Electronic"/ "Electronic pop") to the user who recently transacted items, "Gym Class Heroes" (in class "Pop"/"Dance") and "Blue Foundation" (in class "Electronic"/ "Dance"). The number of recent transactions against "Sam Sparro" is not so many in the training dataset, however, our method can recommend it to the active user. This is because there are several transactions that have classes "Electronic" and "Pop"/"Dance", and item "Sam Sparro". On the other hand, *Time* tends to recommend items that are frequently transacted in the current epoch such as items "NE-YO" and "Blink 182", which were new releases at that time. *Taxonomy* tends to recommend items in the classes that are transacted frequently such as item "Maroon" in class "Alternative rock" and item "Nicki Minaj" in class "R&B", even if they are not in the same classes that the active user has transacted in current epoch. Those results decrease the accuracy of the methods.

6 Conclusion

This paper proposed a novel method that accurately predicts user interests by dividing the historical data into discrete time periods, epochs, and identifying those periods that best match the current transactions of the active user. Moreover, it uses the taxonomy-based approach to model transactions by users and so well identifies the similarity of transactions even if there are few transactions in each epoch. It computes recommendations for the active user by analyzing the extracted transactions. We evaluated our method using a music listening history and a restaurant visit history, and confirmed that our method predicts user interests much more accurately than the previous time-weighted collaborative filtering approach. We also confirmed that our method is superior to the typical topic model when applied to the situations in which items transacted by users dynamically change over time. The basic ideas that underlie in our method are quite simple but have great potential for future recommendation techniques that use the temporal information of the users' item transactions.

Acknowledgements. I dedicate this research study to my wife Kumiko Nakatsuji. She gave me sincere support and encouragement throughout this study.

References

1. Bizer, C., Heath, T., Berners-Lee, T.: Linked Data - The Story So Far. International Journal on Semantic Web and Information Systems 5(3), 1–22 (2009)
2. 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)
3. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent Dirichlet Allocation. Journal of Machine Learning Research 3, 993–1022 (2003)
4. Cantador, I., Castells, P., Bellogín, A.: An Enhanced Semantic Layer for Hybrid Recommender Systems: Application to News Recommendation. Int. J. Semantic Web Inf. Syst. 7(1), 44–78 (2011)
5. Chemudugunta, C., Holloway, A., Smyth, P., Steyvers, M.: Modeling Documents by Combining Semantic Concepts with Unsupervised Statistical Learning. In: Sheth, A.P., Staab, S., Dean, M., Paolucci, M., Maynard, D., Finin, T., Thirunarayan, K. (eds.) ISWC 2008. LNCS, vol. 5318, pp. 229–244. Springer, Heidelberg (2008)
6. Ding, Y., Li, X.: Time weight collaborative filtering. In: Proc. CIKM 2005, pp. 485–492 (2005)
7. Hu, Y., Koren, Y., Volinsky, C.: Collaborative Filtering for Implicit Feedback Datasets. In: Proc. ICDM 2008, pp. 263–272 (2008)
8. Iwata, T., Watanabe, S., Yamada, T., Ueda, N.: Topic tracking model for analyzing consumer purchase behavior. In: Proc. IJCAI 2009, pp. 1427–1432 (2009)
9. Jain, P., Hitzler, P., Sheth, A.P., Verma, K., Yeh, P.Z.: Ontology Alignment for Linked Open Data. 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. 402–417. Springer, Heidelberg (2010)
10. Jamali, M., Ester, M.: Using a trust network to improve top-N recommendation. In: Proc. RecSys 2009, pp. 181–188 (2009)

11. Jech, T.: Set Theory: The Third Millennium Edition, Revised and Expanded. Springer Monographs in Mathematics. Springer (2003)
12. Kolda, T.G., Bader, B.W.: Tensor Decompositions and Applications. SIAM Rev. 51(3), 455–500 (2009)
13. Konstas, I., Stathopoulos, V., Jose, J.M.: On social networks and collaborative recommendation. In: Proc. SIGIR 2009, pp. 195–202 (2009)
14. Koren, Y.: Collaborative filtering with temporal dynamics. In: Proc. KDD 2009, pp. 447–456 (2009)
15. Liu, N.N., Zhao, M., Xiang, E., Yang, Q.: Online evolutionary collaborative filtering. In: Proc. RecSys 2010, pp. 95–102 (2010)
16. Manning, C.D., Raghavan, P., Schtze, H.: Introduction to Information Retrieval. Cambridge University Press (2008)
17. McGuinness, D.L.: Ontologies Come of Age. In: Spinning the Semantic Web, pp. 171–194. MIT Press (2003)
18. Nakatsuji, M., Fujiwara, Y., Tanaka, A., Uchiyama, T., Fujimura, K., Ishida, T.: Classical music for rock fans?: novel recommendations for expanding user interests. In: Proc. CIKM 2010, pp. 949–958 (2010)
19. Nakatsuji, M., Fujiwara, Y., Uchiyama, T., Fujimura, K.: User Similarity from Linked Taxonomies: Subjective Assessments of Items. In: Proc. IJCAI 2011, pp. 2305–2311 (2011)
20. Nakatsuji, M., Miyoshi, Y., Otsuka, Y.: Innovation Detection Based on User-Interest Ontology of Blog Community. 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. 515–528. Springer, Heidelberg (2006)
21. Nakatsuji, M., Yoshida, M., Ishida, T.: Detecting innovative topics based on user-interest ontology. Journal of Web Semantics 7(2), 107–120 (2009)
22. Parameswaran, A.G., Koutrika, G., Bercovitz, B., Garcia-Molina, H.: Recsplorer: recommendation algorithms based on precedence mining. In: Proc. SIGMOD 2010, pp. 87–98 (2010)
23. Parundekar, R., Knoblock, C.A., Ambite, J.L.: Linking and Building Ontologies of Linked Data. 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. 598–614. Springer, Heidelberg (2010)
24. Rendle, S., Freudenthaler, C., Schmidt-Thieme, L.: Factorizing personalized Markov chains for next-basket recommendation. In: Proc. WWW 2010, pp. 811–820 (2010)
25. Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., Riedl, J.: GroupLens: An Open Architecture for Collaborative Filtering of Netnews. In: Proc. CSCW 1994, pp. 175–186 (1994)
26. Rogers, E.M.: Diffusion of innovations, 5th edn. Free Press (2003)
27. Sarwar, B.M., Karypis, G., Konstan, J.A., Riedl, J.: Analysis of recommendation algorithms for e–commerce. In: Proc. EC 2000, pp. 158–167 (2000)
28. Szomszor, M., Alani, H., Cantador, I., O'Hara, K., Shadbolt, N.R.: Semantic Modelling of User Interests Based on Cross-Folksonomy Analysis. In: Sheth, A.P., Staab, S., Dean, M., Paolucci, M., Maynard, D., Finin, T., Thirunarayan, K. (eds.) ISWC 2008. LNCS, vol. 5318, pp. 632–648. Springer, Heidelberg (2008)
29. Xiang, L., Yuan, Q., Zhao, S., Chen, L., Zhang, X., Yang, Q., Sun, J.: Temporal recommendation on graphs via long- and short-term preference fusion. In: Proc. KDD 2010, pp. 723–732 (2010)
30. Ziegler, C.N., McNee, S.M.: Improving recommendation lists through topic diversification. In: Proc. WWW 2005, pp. 22–32 (2005)
31. Zimdars, A., Chickering, D.M., Meek, C.: Using Temporal Data for Making Recommendations. In: Proc. UAI 2001, pp. 580–588 (2001)

Link Discovery with Guaranteed Reduction Ratio in Affine Spaces with Minkowski Measures

Axel-Cyrille Ngonga Ngomo

Department of Computer Science
University of Leipzig
Johannisgasse 26, 04103 Leipzig
ngonga@informatik.uni-leipzig.de
http://bis.uni-leipzig.de/AxelNgonga

Abstract. Time-efficient algorithms are essential to address the complex linking tasks that arise when trying to discover links on the Web of Data. Although several lossless approaches have been developed for this exact purpose, they do not offer theoretical guarantees with respect to their performance. In this paper, we address this drawback by presenting the first Link Discovery approach with theoretical quality guarantees. In particular, we prove that given an achievable reduction ratio r, our Link Discovery approach $\mathcal{HR}^3$ can achieve a reduction ratio $r' \leq r$ in a metric space where distances are measured by the means of a Minkowski metric of any order $p \geq 2$. We compare $\mathcal{HR}^3$ and the HYPPO algorithm implemented in LIMES 0.5 with respect to the number of comparisons they carry out. In addition, we compare our approach with the algorithms implemented in the state-of-the-art frameworks LIMES 0.5 and SILK 2.5 with respect to runtime. We show that $\mathcal{HR}^3$ outperforms these previous approaches with respect to runtime in each of our four experimental setups.

1 Introduction

One of the key principles of the Linked Data paradigm is the inclusion of links between data sets [1]. While this principle is central for tasks such as federated querying [20], cross-ontology question answering [12], large-scale inferences [22] and data integration [3], it is increasingly tedious to implement manually. One of the main difficulty behind the discovery of links is its intrinsic time complexity. Over the last five years, the Linked Data Web has evolved from 12 knowledge bases (May 2007) to more than 295 knowledge bases in September 2011 which contain more than 31 billion triples[1]. The combination of the mere size of these knowledge bases and the quadratic a-priori time complexity of Link Discovery leads to brute-force algorithms requiring weeks and even longer to compute links between large knowledge bases such as DBpedia[2] and LinkedGeoData[3].

[1] http://www4.wiwiss.fu-berlin.de/lodcloud/state/
[2] http://dbpedia.org
[3] http://linkedgeodata.org

Addressing this challenge demands the development of time-efficient and lossless solutions for the computation of links. Link Discovery frameworks such as LIMES [15,14] and SILK [9] have been designed to address this challenge. Yet, none of the manifold approaches they implement provides theoretical guarantees with respect to their performance. Thus, so far, it was impossible to predict how Link Discovery frameworks would perform w.r.t. time or space requirements. Consequently, the deployment of techniques such as customized memory management [5] or time-optimization strategies [23] (e.g., automated scaling for cloud computing when provided with very complex linking tasks) was rendered very demanding if not impossible.

In this paper, we introduce the novel approach $\mathcal{HR}^3$. Similar to the HYPPO algorithm [14] (on whose formalism it is based), $\mathcal{HR}^3$ assumes that the property values that are to be compared are expressed in an affine space with a Minkowski distance. Consequently, it can be most naturally used to process the portion of link specifications that compare numeric values (e.g., temperatures, elevations, populations, etc.). $\mathcal{HR}^3$ goes beyond the state of the art by being able to carry out Link Discovery tasks with *any achievable reduction ratio* [6]. This theoretical guarantee is of practical importance, as it does not only allow our approach to be more time-efficient than the state of the art but also lays the foundation for the implementation of customized memory management and time-optimization strategies for Link Discovery. The three main contributions of this paper are thus as follows:

1. We present a novel indexing scheme for hypercubes in metric spaces with Minkowski distances. This scheme builds the basis upon which $\mathcal{HR}^3$ discards unnecessary comparisons.
2. We prove formally that $\mathcal{HR}^3$'s reduction ratio can be made arbitrarily close to the optimal reduction ratio. For this purpose, we first define the relative reduction ratio (RRR). We then show that $\mathcal{HR}^3$'s RRR converges towards a lower bound and prove this bound to be exactly 1.
3. We show experimentally that in addition to providing theoretical guarantees, our approach outperforms the state of the art. For this purpose, we compare the number of comparisons carried out by $\mathcal{HR}^3$ and HYPPO. In addition, we compare $\mathcal{HR}^3$'s runtime with that of HYPPO (as implemented in LIMES) and SILK[4].

The rest of this paper is structured as follows: In Section 2, we present preliminaries and the notation used to formalize our approach $\mathcal{HR}^3$. We also introduce the relative reduction ratio RRR. We then prove that our algorithm can achieve any RRR score larger than 1 and that we can therewith achieve any possible reduction ratio (Section 3). After a short presentation of the implementation of our algorithm in Section 4, we evaluate our approach against SILK and HYPPO in four experiments in Section 5. Subsequently, in Section 6, we give an overview of previous approaches to Link Discovery. Finally, we discuss our findings and conclude in Section 7.

[4] The algorithm was implemented in the new verison of LIMES, of which a demo is available at http://limes.aksw.org

2 Preliminaries

In this section, we present the preliminaries necessary to understand the subsequent parts of this work. In particular, we define the problem of Link Discovery, the reduction ratio and the relative reduction ratio formally as well as give an overview of space tiling for Link Discovery. The subsequent description of $\mathcal{HR}^3$ relies partly on the notation presented in this section.

2.1 Link Discovery

The goal of Link Discovery is to compute the set of pair of instances $(s,t) \in S \times T$ that are related by a relation R, where S and T are two not necessarily distinct sets of instances. One way to automate this discovery is to compare the $s \in S$ and $t \in T$ based on their properties using a distance measure. Two entities are then considered to be linked via R if their distance is less or equal to a threshold θ [15].

Definition 1 (Link Discovery on Distances). *Given two sets S and T of instances, a distance measure δ over the properties of $s \in S$ and $t \in T$and a distance threshold $\theta \in [0, \infty[$, the goal of Link Discovery is to compute the set $\mathcal{M} = \{(s,t,\delta(s,t)) : s \in S \wedge t \in T \wedge \delta(s,t) \leq \theta\}$.*

Note that in this paper, we are only interested in lossless solutions, i.e., solutions that are able to find all pairs that abide by the definition given above.

2.2 Reduction Ratio

A brute-force approach to Link Discovery would execute a Link Discovery task on S and T by carrying out $|S||T|$ comparisons. One of the key ideas behind time-efficient Link Discovery algorithms $\mathcal{A}$ is to reduce the number of comparisons that are effectively carried out to a number $C(\mathcal{A}) < |S||T|$ [21]. The reduction ratio RR of an algorithm $\mathcal{A}$ is given by

$$RR(\mathcal{A}) = 1 - \frac{C(\mathcal{A})}{|S||T|}. \tag{1}$$

$RR(\mathcal{A})$ captures how much of the Cartesian product $|S||T|$ was not explored before the output of $\mathcal{A}$ was reached. It is obvious that even an optimal lossless solution which performs only the necessary comparisons cannot achieve a RR of 1. Let C_{min} be the minimal number of comparisons necessary to complete the Link Discovery task without losing recall, i.e., $C_{min} = |\mathcal{M}|$. We define the relative reduction ratio $RRR(\mathcal{A})$ as the proportion of the minimal number of comparisons that was carried out by the algorithm $\mathcal{A}$ before it terminated. Formally

$$RRR(\mathcal{A}) = \frac{1 - \frac{C_{min}}{|S||T|}}{1 - \frac{C(\mathcal{A})}{|S||T|}} = \frac{|S||T| - C_{min}}{|S||T| - C(\mathcal{A})}. \tag{2}$$

$RRR(\mathcal{A})$ indicates how close $\mathcal{A}$ is to the optimal solution with respect to the number of candidates it tests. Given that $C(\mathcal{A}) \geq C_{min}$, $RRR(\mathcal{A}) \geq 1$. Note that the larger the value of $RRR(\mathcal{A})$, the poorer the performance of $\mathcal{A}$ with respect to the task at hand.

The main observation that led to this work is that while most algorithms aim to optimize their RR (and consequently their RRR), current approaches to Link Discovery do not provide any guarantee with respect to the RR (and consequently the RRR) that they can achieve. In this work, we present an approach to Link Discovery in metric spaces whose RRR is guaranteed to converge to 1.

2.3 Space Tiling for Link Discovery

Our approach, $\mathcal{HR}^3$, builds upon the same formalism on which the HYPPO algorithm relies, i.e., space tiling. HYPPO addresses the problem of efficiently mapping instance pairs $(s,t) \in S \times T$ described by using exclusively numeric values in a n-dimensional metric space and has been shown to outperform the state of the art in previous work [14]. The observation behind space tiling is that in spaces (Ω, δ) with orthogonal, (i.e., uncorrelated) dimensions[5], common metrics for Link Discovery can be decomposed into the combination of functions $\phi_{i,i\in\{1...n\}}$ which operate on exactly one dimension of $\Omega : \delta = f(\phi_1, ..., \phi_n)$. For Minkowski distances of order p, $\phi_i(x,\omega) = |x_i - \omega_i|$ for all values of i and $\delta(x,\omega) = \sqrt[p]{\sum\limits_{i=1}^{n} \phi_i^p(x,\omega)^p}$. A direct consequence of this observation is the inequality $\phi_i(x,\omega) \leq \delta(x,\omega)$. The basic insight that results this observation is that the hypersphere $H(\omega,\theta) = \{x \in \Omega : \delta(x,\omega) \leq \theta\}$ is a subset of the hypercube V defined as $V(\omega,\theta) = \{x \in \Omega : \forall i \in \{1...n\}, \phi_i(x_i,\omega_i) \leq \theta\}$. Consequently, one can reduce the number of comparisons necessary to detect all elements of $H(\omega,\theta)$ by discarding all elements which are not in $V(\omega,\theta)$ as non-matches. Let $\Delta = \theta/\alpha$, where $\alpha \in \mathbb{N}$ is the *granularity parameter* that controls how fine-grained the space tiling should be (see Figure 1 for an example). We first tile Ω into the adjacent hypercubes (short: cubes) C that contain all the points ω such that

$$\forall i \in \{1...n\}, c_i\Delta \leq \omega_i < (c_i+1)\Delta \text{ with } (c_1, ..., c_n) \in \mathbb{N}^n. \tag{3}$$

We call the vector $(c_1, ..., c_n)$ the coordinates of the cube C. Each point $\omega \in \Omega$ lies in the cube $C(\omega)$ with coordinates $(\lfloor \omega_i/\Delta \rfloor)_{i=1...n}$. Given such a space tiling, it is obvious that $V(\omega,\theta)$ consists of the union of the cubes such that $\forall i \in \{1...n\} : |c_i - c(\omega)_i| \leq \alpha$.

Like most of the current algorithms for Link Discovery, space tiling does not provide optimal performance guarantees. The main goal of this paper is to build upon the tiling idea so as to develop an algorithm that can achieve any possible RR. In the following, we present such an algorithm, $\mathcal{HR}^3$.

[5] Note that in all cases, a space transformation exists that can map a space with correlated dimensions to a space with uncorrelated dimensions.

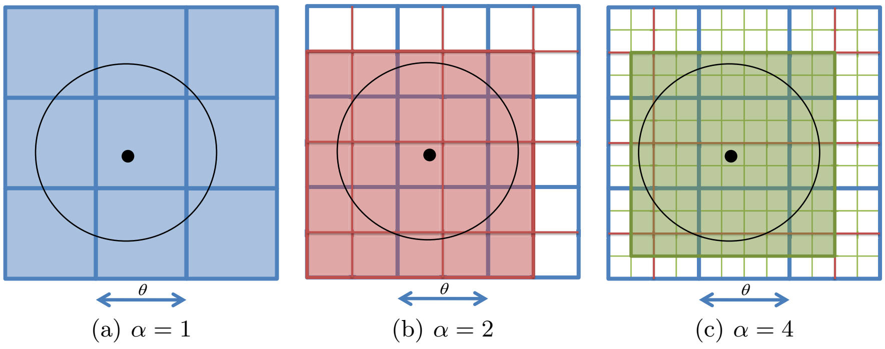


Fig. 1. Space tiling for different values of α. The colored squares show the set of elements that must be compared with the instance located at the black dot. The points within the circle lie within the distance θ of the black dot. Note that higher values of α lead to a better approximation of the hypersphere but also to more hypercubes.

3 Approach

The goal of the $\mathcal{HR}^3$ algorithm is to efficiently map instance pairs $(s,t) \in S \times T$ that are described by using exclusively numeric values in a n-dimensional metric space where the distances are measured by using any Minkowski distance of order $p \geq 2$. To achieve this goal, $\mathcal{HR}^3$ relies on a *novel indexing scheme* that allows achieving any *RRR* greater than or equal to than 1. In the following, we first present our new indexing scheme and show that we can discard more hypercubes than simple space tiling for all granularities α such that $n(\alpha-1)^p > \alpha^p$. We then prove that by these means, our approach can achieve any *RRR* greater than 1, therewith proving the *optimality of our indexing scheme* with respect to *RRR*.

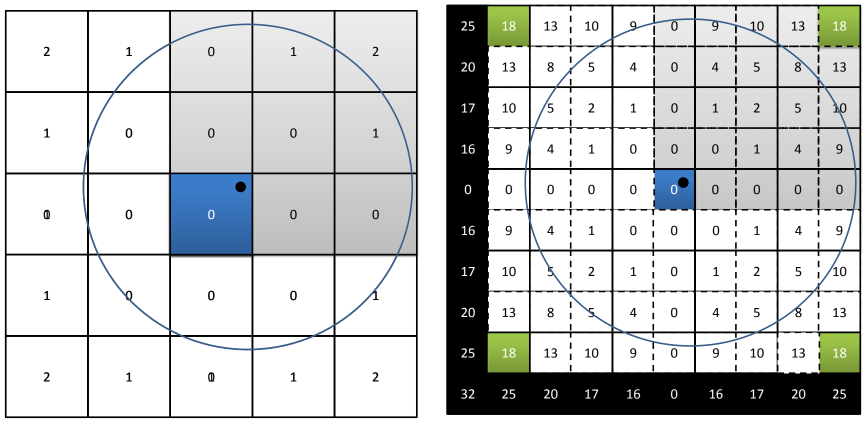
3.1 Indexing Scheme

Let $\omega \in \Omega = S \cup T$ be an arbitrary reference point. Furthermore, let δ be the Minkowski distance of order p. We define the *index* function as follows:

$$index(C,\omega) = \begin{cases} 0 \text{ if } \exists i : |c_i - c(\omega)_i| \leq 1 \text{ with } i \in \{1,...,n\}, \\ \sum\limits_{i=1}^{n} (|c_i - c(\omega)_i| - 1)^p \text{ else,} \end{cases} \quad (4)$$

where C is a hypercube resulting from a space tiling and $\omega \in \Omega$. Figure 2 shows an example of such indexes for $p = 2$ with $\alpha = 2$ (Figure 2(a)) and $\alpha = 4$ (Figure 2(b)).

Note that the blue square with index 0 contains the reference point ω. Also note that our indexing scheme is symmetric with respect to $C(\omega)$. Thus, it is sufficient to prove the subsequent lemmas for hypercubes C such that $c_i > c(\omega)_i$. In Figure 2, this is the upper right portion of the indexed space with the gray



(a) $\alpha = 2$ (b) $\alpha = 4$

Fig. 2. Space tiling and resulting index for a two-dimensional example. Note that the index in both subfigures was generated for exactly the same portion of space. The black dot stands for the position of ω.

background. Finally, note that the maximal index that a hypercube can achieve is $n(\alpha-1)^p$ as $\max|c_i - c_i(\omega)| = \alpha$ per construction of $H(\omega, \theta)$.

The indexing scheme proposed above guarantees the following:

Lemma 1. *$index(C,\omega) = x \rightarrow \forall s \in C(\omega)\ \forall t \in C\ \delta^p(s,t) > x\Delta^p$.*

Proof. This lemma is a direct implication of the construction of the index. $index(C,\omega) = x$ implies that

$$\sum_{i=1}^{n}(c_i - c(\omega)_i - 1)^p = x.$$

Now given the definition of the coordinates of a cube (Eq. 3), the following holds:

$$\forall s \in C(\omega)\ \forall t \in C\ |s_i - t_i| \geq (|c_i - c(\omega)_i| - 1)\Delta.$$

Consequently,

$$\forall s \in C(\omega)\ \forall t \in C\ \sum_{i=1}^{n}|s_i - t_i|^p \geq \sum_{i=1}^{n}(|c_i - c(\omega)_i| - 1)^p\Delta^p.$$

By applying the definition of the Minkowski distance of the index function, we finally get $\forall s \in C(\omega)\ \forall t \in C\ \delta^p(s,t) > x\Delta^p$. □

Note that given that $\omega \in C(\omega)$, the following also holds:

$$index(C,\omega) = x \rightarrow \forall t \in C : \delta^p(\omega,t) > x\Delta^p. \quad (5)$$

3.2 $\mathcal{HR}^3$

The main insight behind $\mathcal{HR}^3$ is that in spaces with Minkowski distances, the indexing scheme proposed above allows to safely (i.e., without dismissing correct matches) discard more hypercubes than when using simple space tiling. More specifically,

Lemma 2. *$\forall s \in S : index(C, s) > \alpha^p$ implies that all $t \in C$ are non-matches.*

Proof. This lemma follows directly from Lemma 1 as

$$index(C, s) > \alpha^p \rightarrow \forall t \in C, \delta^p(s, t) > \Delta^p \alpha^p = \theta^p. \quad (6)$$

For the purpose of illustration, let us consider the example of $\alpha = 4$ and $p = 2$ in the two-dimensional case displayed in Figure 2(b). Lemma 2 implies that any point contained in a hypercube C_{18} with index 18 cannot contain any element t such that $\delta(s, t) \leq \theta$. While space tiling would discard all black cubes in Figure 2(b) but include the elements of C_{18} as candidates, $\mathcal{HR}^3$ discards them and still computes exactly the same results, yet with a better (i.e., smaller) *RRR*.

One of the direct consequences of Lemma 2 is that $n(\alpha - 1)^p > \alpha^p$ is a necessary and sufficient condition for $\mathcal{HR}^3$ to achieve a better *RRR* than simple space tiling. This is simply due to the fact that the largest index that can be assigned to a hypercube is $\sum\limits_{i=1}^{n}(\alpha - 1)^p = n(\alpha - 1)^p$. Now, if $n(\alpha - 1)^p > \alpha^p$, then this cube can be discarded. For $p = 2$ and $n = 2$ for example, this condition is satisfied for $\alpha \geq 4$. Knowing this inequality is of great importance when deciding on when to use $\mathcal{HR}^3$ as discussed in Section 5.

Let $\mathcal{H}(\alpha, \omega) = \{C : index(C, \omega) \leq \alpha^p\}$. $\mathcal{H}(\alpha, \omega)$ is the approximation of the hypersphere $H(\omega) = \{\omega' : \delta(\omega, \omega') \leq \theta\}$ generated by $\mathcal{HR}^3$. We define the volume of $\mathcal{H}(\alpha, \omega)$ as

$$V(\mathcal{H}(\alpha, \omega)) = |\mathcal{H}(\alpha, \omega)|\Delta^p. \quad (7)$$

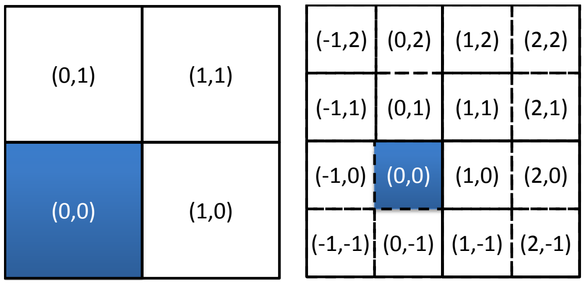
To show that given any $r > 1$, the approximation $\mathcal{H}(\alpha, \omega)$ can always achieve a an $RRR(\mathcal{HR}^3) \leq r$, we begin by showing that

Lemma 3. *$\forall \alpha > 1\ V(\mathcal{H}(\alpha, \omega)) > V(\mathcal{H}(2\alpha, \omega))$.*

Proof. Any cube C discarded by $\mathcal{HR}^3(\alpha)$ is split into 2^n cubes $\mathcal{C}$ by $\mathcal{HR}^3(2\alpha)$, each of which has the coordinates $2c_i$ or $2c_i + 1$. In the worst case for $\mathcal{HR}^3$, ω is assigned the coordinates $2c_i(\omega) + 1$. Figure 3 exemplifies this property of our indexing scheme. Figure 3(b) is an indexing of the same space with the the twofold granularity.

When processed by $\mathcal{HR}^3(2\alpha)$, the minimal index of a hypercube $\mathcal{C}$ is then given by

$$\min index(\mathcal{C}) = \sum_{i=1}^{n}(2c_i - (2c_i(\omega) + 1) - 1)^p = \sum_{i=1}^{n} 2^p(c_i - c(\omega) - 1)^p.$$



(a) Initial coordinates (b) Coordinates with two-fold granularity

Fig. 3. Commparison of coordinates for granularities α and 2α

Given that C was discarded, we know that $\sum\limits_{i=1}^{n}(c_i - c_i(\omega) - 1) > \alpha^p$. Consequently,

$$\min index(\mathcal{C}) > (2\alpha)^p.$$

This leads to all $\mathcal{C}$ that were discarded by $\mathcal{HR}^3(\alpha)$ also being discarded by $\mathcal{HR}^3(2\alpha)$. Proving our lemma is consequently equivalent to showing that there is a hypercube $C' \in \mathcal{H}(\alpha, \omega)$ that is such that one of the 2^n cubes $\mathcal{Q}$ it is split into gets discarded by $\mathcal{HR}^3(2\alpha)$. An example of such a case for $p = 2$ and $n = 2$ is shown in Figures 4(a) and 4(b). For $\alpha = 4$, the cubes that are adjacent to the corner and do not lie on the diagonal of the square are not excluded. Yet, for $\alpha = 8$, 2 of the hypercubes in which they are split are discarded.

Let $C = (c_1, ..., c_n) \notin \mathcal{H}(\omega, \alpha)$ while $C' = (c_1 - 1, ..., c_n) \in \mathcal{H}(\omega, \alpha)$. In the following, we show that the hypercube $\mathcal{Q} = (2c_1 - 1, 2c_2 + 1, ..., 2c_n + 1)$, which is one of the hypercubes that C' gets split into by virtue of its coordinates,[6] will be discarded by $\mathcal{HR}^3(2\alpha)$, i.e., $\mathcal{Q} \notin \mathcal{H}(\omega, 2\alpha)$.

We know that $C = (c_1, ..., c_n)$ gets discarded, i.e., $\sum\limits_{i=1}^{n}(c_i - c_i(\omega) - 1)^p > \alpha^p$.

Now, $\min index(\mathcal{Q}) = (2c_1 - (2c_1(\omega) + 1) - 1)^p + \sum\limits_{i=2}^{n}(2c_i + 1 - (2c_i(\omega) + 1) - 1)^p$.

Consequently, $\min index(\mathcal{Q}) = 2^p(c_1 - c_1(\omega) - 1)^p + \sum\limits_{i=2}^{n}[2(c_i - c_i(\omega) - 1) + 1]^p$.

This value is obviously larger than $\sum\limits_{i=1}^{n}[2(c_i - c_i(\omega) - 1)]^p$. From the premise that $\sum\limits_{i=1}^{n}(c_i - c_i(\omega) - 1)^p > \alpha^p$, we can finally infer that $\min index(\mathcal{Q}) > (2\alpha)^p$. Thus, we can conclude that $\forall \alpha > 1 \; V(\mathcal{H}(\alpha, \omega)) > V(\mathcal{H}(2\alpha, \omega))$. $\square$

One of the consequences of Lemma 2 w.r.t. $RRR(\mathcal{HR}^3, \alpha)$, i.e., the RRR achieved by $\mathcal{HR}^3$ when the granularity is set to α, is

$$\forall \alpha > 1 : \; RRR(\mathcal{HR}^3, \alpha) > RRR(\mathcal{HR}^3, 2\alpha). \tag{8}$$

[6] Note that $2c_1 - 1 = 2(c_1 - 1) + 1$.

Note that this inequality is not sufficient to prove that we can achieve any RRR greater than 1, as series can converge to any real number. Consequently, we still need to show the following:

Lemma 4. $\lim_{\alpha\to\infty} RRR(\mathcal{HR}^3, \alpha) = 1.$

Proof. The cubes that are not discarded by $\mathcal{HR}^3(\alpha)$ are those for which $(|c_i - c_i(\omega)| - 1)^p \leq \alpha^p$. When $\alpha \to \infty$, Δ becomes infinitesimally small, leading to the cubes being single points. Each cube C thus contains a single point x with coordinates $x_i = c_i\Delta$. Especially, $c_i(\omega) = \omega$. Consequently,

$$\sum_{i=1}^{n}(|c_i - c_i(\omega)| - 1)^p \leq \alpha^p \leftrightarrow \sum_{i=1}^{n}\left(\frac{|x_i - \omega_i| - \Delta}{\Delta}\right)^p \leq \alpha^p. \tag{9}$$

Given that $\theta = \Delta\alpha$, we get

$$\sum_{i=1}^{n}\left(\frac{|x_i - \omega_i| - \Delta}{\Delta}\right)^p \leq \alpha^p \leftrightarrow \sum_{i=1}^{n}(|x_i - \omega_i| - \Delta)^p \leq \theta^p. \tag{10}$$

Finally, $\Delta \to 0$ when $\alpha \to \infty$ leads to

$$\sum_{i=1}^{n}(|x_i - \omega_i| - \Delta)^p \leq \theta^p \wedge \alpha \to \infty \to \sum_{i=1}^{n}|x_i - \omega_i|^p \leq \theta^p. \tag{11}$$

This is exactly the condition for linking specified in Definition 1 applied to Minkowski distances of order p. Consequently, $\mathcal{H}(\omega, \infty)$ is exactly $H(\omega, \theta)$ for any θ. Thus, the number of comparisons carried out by $\mathcal{HR}^3(\alpha)$ when $\alpha \to \infty$ is exactly C_{min}, which leads to the conclusion $\lim_{\alpha\to\infty} RRR(\mathcal{HR}^3, \alpha) = 1$. □

Our conclusion is illustrated by Figure 4, which shows the approximations computed by $\mathcal{HR}^3$ for different values of α with $p = 2$ and $n = 2$. The higher α, the closer the approximation is to a circle. Note that these results allow to conclude that for any RRR-value r larger than 1, there is a setting of $\mathcal{HR}^3$ that can compute links with a RRR smaller or equal to r.

4 Implementation

The $\mathcal{HR}^3$ algorithm was implemented as shown in Algorithm 1. It is important to notice that the memory requirements of $\mathcal{HR}^3$ are smaller than those of most other approaches and especially than those of simple space tiling for any α such that $n(\alpha - 1)^p > \alpha^p$, as $\mathcal{HR}^3$ then generates less hypercubes. Yet, $\mathcal{HR}^3$ requires one supplementary computational step as it has to compute the index of cubes before discarding the unnecessary ones. Consequently, although we have shown that $\mathcal{HR}^3$ can achieve any $RRR > 1$, the question that remains to elucidate is whether this theoretical guarantee also offers a practically superior algorithm w.r.t. its runtime. That is the goal of the subsequent evaluation.

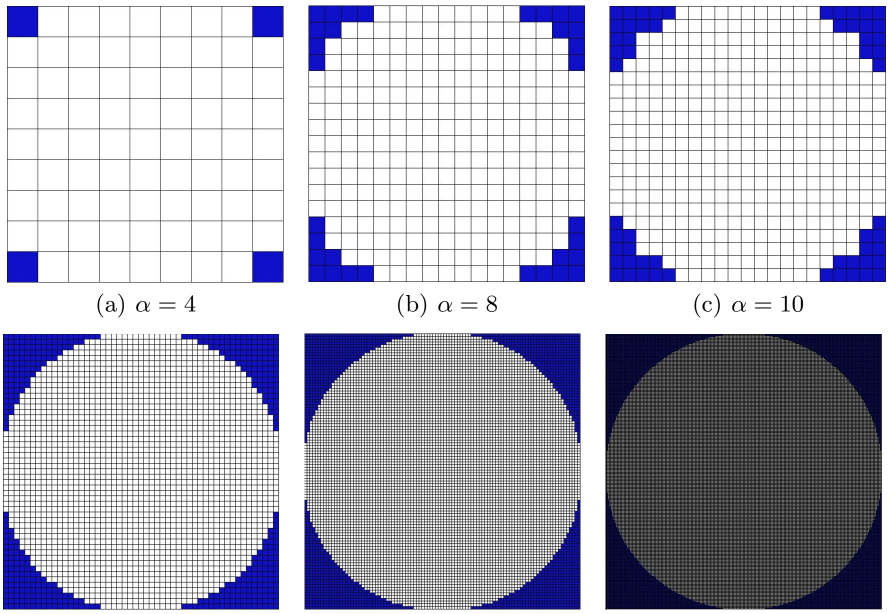

(a) $\alpha = 4$ (b) $\alpha = 8$ (c) $\alpha = 10$
(d) $\alpha = 25$ (e) $\alpha = 50$ (f) $\alpha = 100$

Fig. 4. Approximation generated by $\mathcal{HR}^3$ for different values of α. The white squares are selected whilst the colored ones are discarded.

5 Evaluation

5.1 Experimental Setup

We carried out four experiments to compare $\mathcal{HR}^3$ with LIMES 0.5's HYPPO and SILK 2.5.1. In the first and second experiments, we aimed to deduplicate DBpedia places by comparing their names (`rdfs:label`), minimum elevation, elevation and maximum elevation. We retrieved 2988 entities that possessed all four properties. We use the Euclidean metric on the last three values with the thresholds 49 meters resp. 99 meters for the first resp. second experiment. The third and fourth experiments aimed to discover links between Geonames and LinkedGeoData. Here, we compared the labels (`rdfs:label`), longitude and latitude of the instances. This experiment was of considerably larger scale than the first one, as we compared 74458 entities in Geonames with 50031 entities from LinkedGeoData. Again, we measured the runtime necessary to compare the numeric values when comparing them by using the Euclidean metric. We set the distance thresholds to 1 resp. 9° in experiment 3 resp. 4. We ran all experiments on the same Windows 7 Enterprise 64-bit computer with a 2.8GHz i7 processor with 8GB RAM. The JVM was allocated 7GB RAM to ensure that the runtimes were not influenced by swapping. Only one of the kernels of the processors was used. Furthermore, we ran each of the experiments three times and report the best runtimes in the following.

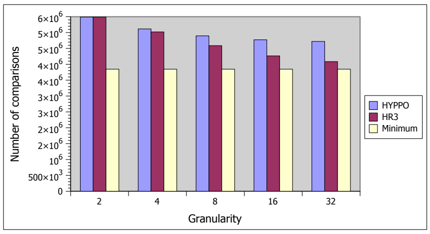
Algorithm 1. The $\mathcal{HR}^3$ algorithm

```
Require: Source knowledge base $S$, target knowledge base $T$, distance threshold $\theta$,
  Minkowski distance $\delta$ of order $p$, granularity factor $\alpha$
  Mapping $M := \emptyset$
  $\Delta = \theta/\alpha$
  for $\omega \in S \cup T$ do
    $C(\lfloor\omega_1/\Delta\rfloor, ..., \lfloor\omega_n/\Delta\rfloor) := C(\lfloor\omega_1/\Delta\rfloor, ..., \lfloor\omega_n/\Delta\rfloor) \cup \{\omega\}$
  end for
  for $s \in S$ do
    for $C \in \mathcal{H}(s, \alpha)$ do
      for $t \in C \cap T$ do
        if $\delta(s,t) \leq \theta$ then
          $M := M \cup (s, t, \delta(s,t))$
        end if
      end for
    end for
  end for
  return $M$
```

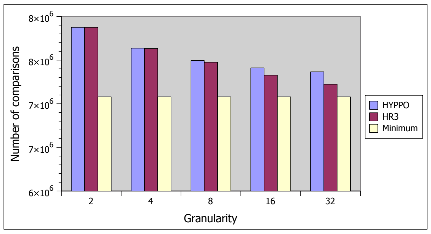
5.2 Results

We first measured the number of comparisons required by HYPPO and $\mathcal{HR}^3$ to complete the tasks at hand (see Figure 5). Note that we could not carry out this section of the evaluation for SILK2.5.1 as it would have required altering the code of the framework. In the experiments 1, 3 and 4, $\mathcal{HR}^3$ can reduce the overhead in comparisons (i.e., the number of unnecessary comparisons divided by the number of necessary comparisons) from approximately 24% for HYPPO to approximately 6% (granularity = 32). In experiment 2, the overhead is reduced from 4.1% to 2%. This difference in overhead reduction is mainly due to the data clustering around certain values and the clusters having a radius between 49 meters and 99 meters. Thus, running the algorithms with a threshold of 99 meters led to only a small a-priori overhead and HYPPO performing remarkably well. Still, even on such data distributions, $\mathcal{HR}^3$ was able to discard even more data and to reduce the number of unnecessary computations by more than 50% relative. In the best case (Exp. 4, $\alpha = 32$, see Figure 5(d)), $\mathcal{HR}^3$ required approximately 4.13×10^6 less comparisons than HYPPO for $\alpha = 32$. Even for the smallest setting (Exp. 1, see Figure 5(a)), $\mathcal{HR}^3$ still required 0.64×10^6 less comparisons.

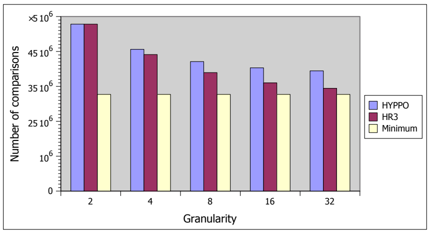
We also measured the runtimes of SILK, HYPPO and $\mathcal{HR}^3$. The best runtimes of the three algorithms for each of the tasks is reported in Figure 6. Note that SILK's runtimes were measured without the indexing time, as the data fetching and indexing are merged to one process in SILK. Also note that in the second experiment, SILK did not terminate due to higher memory requirements. We approximated SILK's runtime by extrapolating approximately 11 min it required for 8.6% of the computation before the RAM was filled. Again, we did not consider the indexing time.



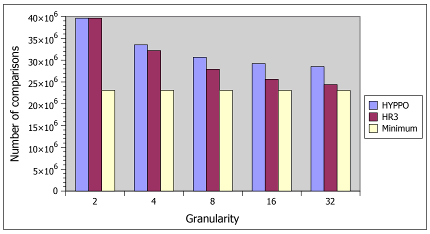
(a) Experiment 1



(b) Experiment 2



(c) Experiment 3



(d) Experiment 4

Fig. 5. Number of comparisons for $\mathcal{HR}^3$ and HYPPO

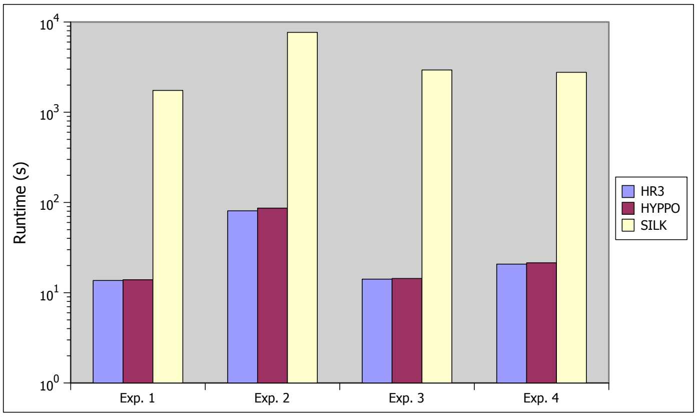
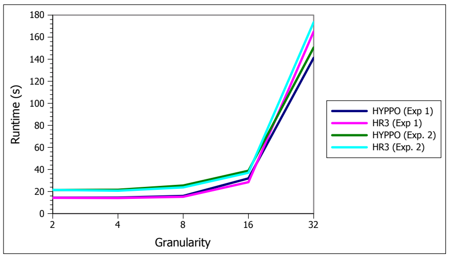


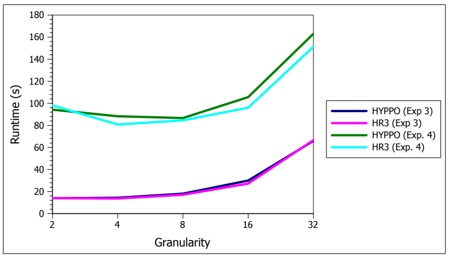
Fig. 6. Comparison of the runtimes of $\mathcal{HR}^3$, HYPPO and SILK2.5.1

Due to the considerable difference in runtime (approximately 2 orders of magnitude) between HYPPO and SILK, we report solely HYPPO and $\mathcal{HR}^3$'s runtimes in the detailed runtimes figures 7(a) and 7(b). Overall, $\mathcal{HR}^3$ outperformed the other two approaches in all experiments, especially for $\alpha = 4$. It is important to note that the improvement in runtime increases with the complexity of the experiment. For example, while $\mathcal{HR}^3$ outperforms HYPPO by 3% in the second experiment, the different grows to more than 7% in the fourth experiment. In addition, the improvement in runtime augments with the threshold. This can be seen in the third and fourth experiments. While $\mathcal{HR}^3$ is less than 2% faster in the third experiment, it is more than 7% faster when $\theta = 4$ the fourth experiment . As expected, $\mathcal{HR}^3$ is slower than HYPPO for $\alpha < 4$ as it carries out exactly the same comparisons but still has the overhead of computing the index. Yet, given that we know that $\mathcal{HR}^3$ is only better when $n(\alpha - 1)^p > \alpha^p$, our implementation only carries out the indexing when this inequality holds. By these means, we can ensure that $\mathcal{HR}^3$ is only used when it is able to discard hypercubes that HYPPO would not discard, therewith reaching superior runtimes both with small and large values α. Note that the difference between the improvement of the number of comparisons necessitated by $\mathcal{HR}^3$ and the improvement in runtime over all experiments is due to the supplementary indexing step required by $\mathcal{HR}^3$.

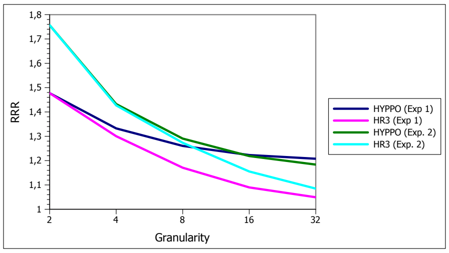
Finally, we measured the RRR of both $\mathcal{HR}^3$ and HYPPO (see Figures 7(c) and 7(d)). In the two-dimensional experiments 3 and 4, HYPPO achieves a RRR close to 1. Yet, it is still outperformed by $\mathcal{HR}^3$ as expected. A larger difference between the RRR of $\mathcal{HR}^3$ and HYPPO can be seen in the three-dimensional experiments, where the RRR of both algorithms diverge significantly. Note that the RRR difference grows not only with the number of dimensions but also with the size of the problem. The difference in RRR between HYPPO and $\mathcal{HR}^3$ does not always reflect the difference in runtime due to the indexing overhead of $\mathcal{HR}^3$. Still, for $\alpha = 4$, $\mathcal{HR}^3$ generates a sufficient balance of indexing runtime and comparison runtime (i.e., RRR) to outperform HYPPO in all experiments.



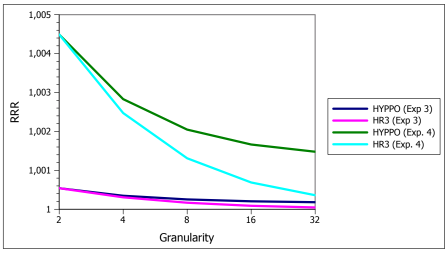
(a) Runtimes for experiments 1 and 2



(b) Runtimes for experiments 3 and 4



(c) RRR for experiments 1 and 2



(d) RRR for experiments 3 and 4

Fig. 7. Comparison of runtimes and RRR of $\mathcal{HR}^3$ and HYPPO

6 Related Work

The growing size and number of knowledge bases available in the Linked Data Cloud makes Link Discovery intrinsically complex with respect to its runtime. To address this issue, manifold time-efficient frameworks have been developed. LIMES [14] offers a complex grammar for link specifications that can be translated into a combination of time-efficient atomic mappers that are combined via a hybrid approach. For example, LIMES implements a dedicated approach for numeric values called HYPPO. SILK [9] implements a different Link Discovery paradigm and aims to place all instances that are to be compared in a multi-dimensional space. It then uses MultiBlock to discard unnecessary comparisons efficiently. In contrast to LIMES and SILK, which implement lossless approaches, the approach presented in [21] uses a candidate selection approach based on discriminative properties to compute links very efficiently but potentially loses links while doing so. Other frameworks and approaches include those described in [18,8,19].

Albeit Link Discovery is closely related with record linkage [7] and deduplication [4], it is important to notice that Link Discovery goes beyond these two tasks as Link Discovery aims to provide the means to link entities via arbitrary relations. Different blocking techniques such as standard blocking, sorted-neighborhood, bi-gram indexing, canopy clustering and adaptive blocking have been developed by the database community to address the problem of the quadratic time complexity of brute force comparison [11]. In addition, very time-efficient approaches have been proposed to compute string similarities for record linkage, including All-Pairs [2], PPJoin and PPJoin+ [24]. However, these approaches alone cannot deal with the diversity of property values found on the Web of Data as they cannot deal with numeric values. In addition, most time-efficient string matching algorithms can only deal with simple link specifications, which are mostly insufficient when computing links between large knowledge bases.

In recent work, the discovery of adequate link specifications has been addressed mainly by using machine learning approaches. For example, [21] detect discriminative properties by using string concatenations. RAVEN [16] combines stable marriage algorithms and a perceptron-based learning algorithm with the frame of active learning to compute boolean and linear classifiers. SILK [10] employs genetic programming to learn link configurations from positive and negative examples. [13] go a step further and combine genetic programming with active learning to discover high-accuracy link specificity with a small number of annotations. Another approach based on genetic programming is presented in [17]. Here, the authors show how link specifications can be learned without any input from the user. To the best of our knowledge, none of the approaches presented previously provide formal guarantees w.r.t. their performance. $\mathcal{HR}^3$ is the first matching approach that it guaranteed not to lose links while converging to the small possible reduction ratio. Note that while $\mathcal{HR}^3$ was designed for numeric values, it can be used in any space with Minkowski distances, for example for comparing indexes in multi-dimensional spaces. Thus, it can be used for any datatype mapped to a metric space.

7 Conclusion and Future Work

In this paper, we presented $\mathcal{HR}^3$, a time-efficient approach for the discovery of links in spaces with Minkowski distances. We proved that our approach can achieve is optimal w.r.t its reduction ration by showing that its RRR converges towards 1 when α converges towards ∞. It is important to note that an optimal $RRR(\mathcal{A})$ does not necessarily mean that $\mathcal{A}$ outperforms algorithms with a poorer RRR with respect to runtime as achieving a good RRR score usually requires better preprocessing (usually in form of indexing), which might be more time-demanding than the combination of a rougher preprocessing and a run with a poorer RRR. Thus, in addition to proving formally that we can guarantee a RRR that converges towards 1, we implemented our approach and compared it with the state-of-the-art algorithms HYPPO implemented in LIMES and SILK. We showed that we outperform both frameworks w.r.t. to their runtime and that we reach RRR close to 1 for α as small as 32. Our experiments also showed that $\alpha = 4$ is a good setup for $\mathcal{HR}^3$. Our approach aims to be the first of a novel type of Link Discovery approaches, i.e., of approaches which can guarantee theoretical optimality while also being empirically usable. In future work, we will thus aim to develop more of such approaches and to make use of their theoretical characteristics for memory and space management. With respect to $\mathcal{HR}^3$, we will mainly improve the implementation of its indexing to ensure even better runtimes.

References

1. Auer, S., Lehmann, J., Ngonga Ngomo, A.-C.: Introduction to Linked Data and Its Lifecycle on the Web. In: Polleres, A., d'Amato, C., Arenas, M., Handschuh, S., Kroner, P., Ossowski, S., Patel-Schneider, P. (eds.) Reasoning Web 2011. LNCS, vol. 6848, pp. 1–75. Springer, Heidelberg (2011)
2. Bayardo, R.J., Ma, Y., Srikant, R.: Scaling up all pairs similarity search. In: WWW, pp. 131–140 (2007)
3. Ben-David, D., Domany, T., Tarem, A.: Enterprise Data Classification Using Semantic Web Technologies. 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. 66–81. Springer, Heidelberg (2010)
4. Bleiholder, J., Naumann, F.: Data fusion. ACM Comput. Surv. 41(1), 1–41 (2008)
5. Botelho, F.C., Ziviani, N.: External perfect hashing for very large key sets. In: CIKM, pp. 653–662 (2007)
6. Elfeky, M.G., Elmagarmid, A.K., Verykios, V.S.: Tailor: A record linkage tool box. In: ICDE, pp. 17–28 (2002)
7. Elmagarmid, A.K., Ipeirotis, P.G., Verykios, V.S.: Duplicate record detection: A survey. IEEE Transactions on Knowledge and Data Engineering 19, 1–16 (2007)
8. Glaser, H., Millard, I.C., Sung, W.-K., Lee, S., Kim, P., You, B.-J.: Research on linked data and co-reference resolution. Technical report, University of Southampton (2009)
9. Isele, R., Jentzsch, A., Bizer, C.: Efficient Multidimensional Blocking for Link Discovery without losing Recall. In: WebDB (2011)

10. Isele, R., Bizer, C.: Learning Linkage Rules using Genetic Programming. In: Sixth International Ontology Matching Workshop (2011)
11. Köpcke, H., Thor, A., Rahm, E.: Comparative evaluation of entity resolution approaches with fever. Proc. VLDB Endow. 2(2), 1574–1577 (2009)
12. Lopez, V., Uren, V., Sabou, M.R., Motta, E.: Cross ontology query answering on the semantic web: an initial evaluation. In: K-CAP 2009, pp. 17–24. ACM, New York (2009)
13. Ngonga Ngomo, A.-C., Lyko, K.: EAGLE: Efficient Active Learning of Link Specifications Using Genetic Programming. In: Simperl, E., Cimiano, P., Polleres, A., Corcho, O., Presutti, V. (eds.) ESWC 2012. LNCS, vol. 7295, pp. 149–163. Springer, Heidelberg (2012)
14. Ngonga Ngomo, A.-C.: A Time-Efficient Hybrid Approach to Link Discovery. In: Sixth International Ontology Matching Workshop (2011)
15. Ngonga Ngomo, A.-C., Auer, S.: LIMES - A Time-Efficient Approach for Large-Scale Link Discovery on the Web of Data. In: Proceedings of IJCAI (2011)
16. Ngonga Ngomo, A.-C., Lehmann, J., Auer, S., Höffner, K.: RAVEN – Active Learning of Link Specifications. In: Proceedings of OM@ISWC (2011)
17. 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)
18. Raimond, Y., Sutton, C., Sandler, M.: Automatic interlinking of music datasets on the semantic web. In: Proceedings of the 1st Workshop about Linked Data on the Web (2008)
19. Scharffe, F., Liu, Y., Zhou, C.: Rdf-ai: an architecture for rdf datasets matching, fusion and interlink. In: Proc. of IJCAI IR-KR Workshop (2009)
20. 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)
21. 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)
22. Urbani, J., Kotoulas, S., Maassen, J., van Harmelen, F., Bal, H.: OWL Reasoning with WebPIE: Calculating the Closure of 100 Billion Triples. 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. 213–227. Springer, Heidelberg (2010)
23. Vaquero, L.M., Rodero-Merino, L., Buyya, R.: Dynamically scaling applications in the cloud. SIGCOMM Comput. Commun. Rev. 41, 45–52
24. Xiao, C., Wang, W., Lin, X., Yu, J.X.: Efficient similarity joins for near duplicate detection. In: WWW, pp. 131–140 (2008)

Hitting the Sweetspot: Economic Rewriting of Knowledge Bases

Nadeschda Nikitina[1] and Birte Glimm[2]

[1] Karlsruhe Institute of Technology, Institute AIFB, DE
[2] University of Ulm, Institute of Artificial Intelligence, DE

Abstract. Three conflicting requirements arise in the context of knowledge base (KB) extraction: the size of the extracted KB, the size of the corresponding signature and the syntactic similarity of the extracted KB with the original one. Minimal module extraction and uniform interpolation assign an absolute priority to one of these requirements, thereby limiting the possibilities to influence the other two. We propose a novel technique for $\mathcal{EL}$ that does not require such an extreme prioritization. We propose a tractable rewriting approach and empirically compare the technique with existing approaches with encouraging results.

1 Introduction

In view of the practical deployment of the W3C-specified OWL Web Ontology Language [9] and its specific tractable sublanguages (the so-called *profiles* [5]), non-standard reasoning services supporting different ontology engineering tasks for lightweight logics have gained in importance. Amongst others, the task of semantics-preserving knowledge base extraction for a particular subset of terms has been investigated by the research community: given a knowledge base using a certain vocabulary (called a signature), and a subset of "relevant terms" of that vocabulary, find a knowledge base that contains as little irrelevant information as possible, and, at the same time, contains all information about the relevant terms.

Among the applications of knowledge base extraction is ontology reuse, which helps reducing the expenses of knowledge intensive applications by exploiting the variety of the existing large ontologies. Since the size of a knowledge base has a crucial impact on the maintenance costs and often on the performance of reasoning, it is important to keep the corresponding knowledge base as compact as possible. Knowledge base extraction ideally reduces the amount of irrelevant information imported from external sources, and, at the same time, preserves all relevant consequences. Another application is supporting knowledge engineers in modeling a particular domain or in understanding existing models by revealing dependencies between particular concepts and roles, as, for instance, in the case of interactive ontology revision [8]. Due to its usefulness in various contexts, the task of knowledge base extraction has been investigated by different authors. The currently existing semantics-preserving approaches can be divided into those that compute a subset of the original ontology entailing all relevant consequences (module extraction), e.g., [3,1], and those rewriting the original ontology to contain only relevant terms while preserving all relevant consequences (uniform interpolation), e.g., [2,4,7]. The complexity results for approaches computing a minimal solution are not very promising: even for the lightweight logic $\mathcal{EL}$, the task of minimal module extraction is EXPTIME-hard and the task of uniform interpolation is even 3-EXPTIME-hard with a tight triple-exponential bound on the size of uniform interpolants in case a finite result exists [7]. Given that most applications of knowledge base extraction are of particular interest for large ontologies and that there are scenarios, in which long computation times are not feasible due to user interaction, tractable approaches computing a small but not necessarily minimal solution would often be a reasonable alternative. Moreover, both types of approaches are based on a specific prioritization of objectives that might be necessary in particular scenarios, but is disadvantageous in many others due to its negative impact on the size of the extracted knowledge bases.

In this paper, we consider three conflicting objectives for knowledge base extraction: reducing the size of the extracted knowledge base, reducing the size of its signature and preserving the syntactic similarity of the extracted knowledge base with the originally given one. We demonstrate that, both, minimal module extraction and uniform interpolation, assign an absolute priority to one of these objectives, thereby limiting the possibilities to achieve an improvement w.r.t. the other two. While minimal module extraction only considers subsets of the original knowledge base, thereby requiring a very strong notion of syntactic similarity, uniform interpolation fixes the signature of the extracted knowledge base, possibly yielding triple-exponentially many double-exponentially large axioms. To address scenarios, where the above uncompromising prioritization is not required, we investigate alternative prioritization, allowing for a more balanced relationship between the extents to which the objectives are achieved.

We consider the task of knowledge base extraction for the lightweight logic $\mathcal{EL}$ based on two alternative, less restrictive notions of structural similarity, further assigning the second-highest priority to the knowledge base size. First, we discuss the extraction of knowledge bases consisting only of sub-expressions occurring in the original knowledge base. We give a polynomially-bounded rewriting making particular simple consequences within the knowledge base explicit, such that minimal modules meeting this similarity requirement can be obtained in EXPTIME by applying minimal module extraction to the extended knowledge base.

Second, we consider the extraction of knowledge bases that consist of concepts structurally equivalent to sub-expressions occurring in the original knowledge base, i.e., concepts with the same structure but possibly a different set of atomic concepts. While the extraction of such minimal knowledge bases by first extending the knowledge base and then applying minimal module extraction requires in the worst-case double-exponential time, we propose a tractable rewriting approach that aims at obtaining small but not necessarily minimal knowledge bases. The approach is based on the same elementary rewriting operation as uniform interpolation in [7], namely replacing atomic concepts within expressions by their subsumees and subsumers. However, in order to obtain polynomial bounds and preserve the required structural similarity, we impose additional restrictions on the rewriting, excluding elementary rewriting operations with a negative effect on the module size or structure.

As we show in our evaluation using the Gene Ontology, knowledge bases obtained by our approach on average contain half as many axioms as their minimal justifications within the original knowledge base. A comparison with the existing implementations also yields promising results. In case of the minimal module extractor for DL-Lite$_{\texttt{bool}}$, the extracted modules are 2 to 2.2 times larger than the knowledge bases obtained by our approach. The locality-based module extractor, which is a tractable approach for extracting small but not necessarily minimal subsets of an ontology, extracts modules that are on average 12 times larger than the knowledge bases obtained by our approach.

The paper is organized as follows: In Section 2, we recall the necessary preliminaries on $\mathcal{EL}$. Section 3 formally introduces the task of knowledge base extraction and discusses the conflicting objectives for this task. In Section 4, we show how minimal modules meeting the corresponding requirements of syntactic similarity can be obtained using minimal module extraction. In Sections 5 and 6, we propose a tractable alternative for minimal module extraction based on rewriting. After introducing rewriting in Section 5, in Section 6 we discuss the necessary restrictions on rewriting operations in order to obtain polynomial bounds and preserve the required structural similarity. Finally, we present the evaluation results in Section 7 before we conclude in Section 8. Further details and proofs can be found in the extended version of this paper [6].

2 Preliminaries

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 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 $D \sqsubseteq C$. While knowledge bases in general can also include a specification of individuals with the corresponding concept and role assertions (ABox), in this paper we abstract from ABoxes and concentrate on TBoxes. The signature of an $\mathcal{EL}$ concept C or an axiom α, denoted by sig(C) or sig(α), 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 sig$_C(C)$ and sig$_R(C)$, respectively. The signature of a TBox $\mathcal{T}$, in symbols sig$(\mathcal{T})$ (correspondingly, sig$_C(\mathcal{T})$ and sig$_R(\mathcal{T})$), is defined analogously. Next, we recall the semantics of the above introduced DL constructs, which is defined by the 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 $\mathcal{T}$.

3 Knowledge Base Extraction Revisited

While, in principle, there exist many approaches without a logical background, in this work we focus on logic-based approaches, i.e., approaches that guarantee a preservation of the semantics for the set of relevant entities. We say that the semantics is preserved, if all logical consequences concerning only the relevant entities are preserved. The logical foundation for such a preservation of relevant consequences is given by the established notion of *inseparability*. Two knowledge bases, $\mathcal{T}_1$ and $\mathcal{T}_2$, are inseparable w.r.t. a signature Σ if they have the same Σ-consequences, i.e., consequences whose signature is a subset of Σ. Depending on the particular application requirements, the expressivity of those Σ-consequences can vary from subsumption queries and instance queries to conjunctive queries. In the following, we consider concept inseparability of general $\mathcal{EL}$ terminologies defined analogously to previous work [3,2,4,7], as follows:

Definition 1. *Let $\mathcal{T}_1$ and $\mathcal{T}_2$ be two general $\mathcal{EL}$ knowledge bases and Σ a signature. $\mathcal{T}_1$ and $\mathcal{T}_2$ are concept-inseparable w.r.t. Σ, in symbols $\mathcal{T}_1 \equiv^c_\Sigma \mathcal{T}_2$, if for all $\mathcal{EL}$ concepts C, D with sig$(C) \cup$ sig$(D) \subseteq \Sigma$ holds $\mathcal{T}_1 \models C \sqsubseteq D$, iff $\mathcal{T}_2 \models C \sqsubseteq D$.*

Given a signature Σ and a knowledge base $\mathcal{T}$, the task of knowledge base extraction in general is to compute a knowledge base $\mathcal{T}'$, which is entailed by $\mathcal{T}$ and is concept-inseparable from it. We call the result $\mathcal{T}'$ a *general module* of $\mathcal{T}$.

Definition 2. *Let $\mathcal{T}$ be an $\mathcal{EL}$ knowledge base and Σ a signature. An $\mathcal{EL}$ knowledge base $\mathcal{T}'$ is a* general module *of $\mathcal{T}$ w.r.t. Σ, written $\mathcal{T}' \in \mathtt{MOD}(\mathcal{T}, \Sigma)$, iff (1) $\mathcal{T} \equiv^c_\Sigma \mathcal{T}'$ and (2) $\mathcal{T} \models \mathcal{T}'$.*

The above definition is very generic. It captures the preservation of the semantics, but does not address the quality criteria important for general modules in order to be useful in practice. We consider the following requirements for the task of knowledge base extraction:

1. **Syntactic Similarity**: In scenarios, where the knowledge base is meant to be used by human experts, the syntactic structure of the module determining its comprehensiveness or cognitive complexity has to be taken into account. The extent, to which a general module has to be syntactically similar to the original knowledge base $\mathcal{T}$ depends on the particular application requirements. For instance, modules can be required to be a subset of $\mathcal{T}$, to consist only of sub-expressions occurring in $\mathcal{T}$ or to consist only of concepts structurally equivalent to sub-expressions occurring in $\mathcal{T}$, but possibly referencing different atomic concepts.
2. **Small Knowledge Base Size**: Reducing the size of the knowledge base is a core objective for the task of knowledge base extraction, since smaller knowledge bases (assuming that the particular syntactic similarity requirement is fulfilled in both cases) require less computational and manual effort in many different ontology management activities.
3. **Small Signature Size**: Decreasing the size of the signature results in a decrease of irrelevant entities occurring in the knowledge base, which is also one of the core objectives of knowledge base extraction.

While uniform interpolation clearly prioritizes small signature size making no compromises w.r.t. the other two requirements, minimal module extraction gives the highest priority to syntactic similarity, thereby not allowing for rewriting and, therefore, limiting the possibilities to reduce the size. While such uncompromising prioritization can be required in some particular scenarios, in other scenarios it leads to a disadvantage. The following example demonstrates the drawbacks of minimal module extraction and uniform interpolation in terms of knowledge base size caused by the extreme choice of priorities.

Example 1. Consider the following knowledge base $\mathcal{T}$:[1]

$$A_{13} \sqsubseteq A_9 \quad A_{10} \sqcap A_{13} \sqsubseteq A_{16} \quad A_9 \sqsubseteq \exists r.A_9 \quad A_{i+1} \sqsubseteq A_i \qquad 1 \leq i \leq 14, i \neq 13$$

For the signature $\Sigma = \{A_1, A_8, A_{12}, A_{15}, A_{16}, r\}$, neither the uniform interpolation nor minimal module extraction are effective in terms of reducing the size. While minimal module extraction would return the whole knowledge base, uniform interpolation fails to extract a finite knowledge base due to the cyclic dependency given by $A_9 \sqsubseteq \exists r.A_9$. However, if we are not restricted to subsets of $\mathcal{T}$, but are also interested in modules consisting of sub-expressions occurring in $\mathcal{T}$, then there is a representation of the relevant information about Σ, which uses half as many axioms as the original TBox: $\{A_{12} \sqsubseteq A_{10}$, $A_{15} \sqsubseteq A_{13}$, $A_{10} \sqcap A_{13} \sqsubseteq A_{16}$, $A_{10} \sqsubseteq A_9$, $A_{13} \sqsubseteq A_9$, $A_9 \sqsubseteq \exists r.A_9$, $A_9 \sqsubseteq A_8$, $A_8 \sqsubseteq A_1\}$. If we are, additionally, allowed to exchange atomic concepts within sub-expressions while leaving the structure of expressions unchanged, then there is an even smaller representation consisting of 6 axioms: $\{A_{12} \sqcap A_{15} \sqsubseteq A_{16}$, $A_{12} \sqsubseteq A_9$, $A_{15} \sqsubseteq A_9$, $A_8 \sqsubseteq A_1$, $A_9 \sqsubseteq A_8$, $A_9 \sqsubseteq \exists r.A_9\}$.

In the following, we aim at establishing a balance between these requirements in order to account for scenarios not requiring the above mentioned extreme prioritization. The following example completes the picture roughly sketched above and demonstrates mutual influences of the three requirements upon each other.

Example 2. The following TBox $\mathcal{T}$ models a "counter" with numbers $X_0, \ldots, X_{10}$, where the lowest number X_0 has two subsumees:

$$A_1 \sqsubseteq X_0 \quad A_2 \sqsubseteq X_0 \quad \exists r.X_i \sqcap \exists s.X_i \sqsubseteq X_{i+1} \qquad 0 \leq i \leq 9$$

Given this TBox, we could extract a knowledge base not referencing a particular atomic concept by replacing its occurrence by its direct subsumees. For instance, if we want to represent the information without using X_1, we can omit $\exists r.X_0 \sqcap \exists s.X_0 \sqsubseteq X_1$ and replace X_1 on the left-hand side of the remaining axioms by its direct subsumee $\exists r.X_0 \sqcap \exists s.X_0$, leading to $\exists r.(\exists r.X_0 \sqcap \exists s.X_0) \sqcap \exists s.(\exists r.X_0 \sqcap \exists s.X_0) \sqsubseteq X_2$. Concerning the extraction of knowledge bases from $\mathcal{T}$, we can more generally observe the following:

- Assume that we are interested in the dependencies between X_0, and X_{10} including those using roles r, s. By replacing any of the concepts $X_1, ..., X_9$ by their direct subsumees, we reduce both, the number of axioms and the number of referenced concept names, but we increase the nesting depth of the resulting TBox. A complete replacement of $X_1, ..., X_9$ would yield a subsumee of X_{10} with a nesting depth of 10 and exponentially many occurrences of X_0. Even though the TBox contains only three axioms and no irrelevant concept names, it is less comprehensive than the original knowledge base.

[1] The TBox is structurally similar to minimal modules obtained within our evaluation and can be extended to a more typical TBox by adding more axioms to the subsumption hierarchy without any effect on the obtained general modules.

– Assume that we are interested in A_1, A_2 instead of X_0. Eliminating X_0 from $\mathcal{T}$ would yield four different subsumees of X_1, namely $\exists r.A_1 \sqcap \exists s.A_1$, $\exists r.A_1 \sqcap \exists s.A_2$, $\exists r.A_2 \sqcap \exists s.A_1$ and $\exists r.A_2 \sqcap \exists s.A_2$. Each of these subsumees is required in order to preserve the relevant consequences, since none of the four concepts subsumes another. Replacing X_0 in the extracted knowledge base using only A_1, A_2, X_0, X_{10} and r, s by its two subsumees, A_1 and A_2, would result in double exponentially many ($2^{2^{10}}$) different subsumees of X_{10}. Therefore, the elimination of a single concept name is, in most cases, not justified from the practical point of view.

While Example 1 focuses on the disadvantage in terms of knowledge base size caused by an unnecessarily strong notion of syntactic similarity, the latter example demonstrates more clearly the effect of unrestricted rewriting aiming at signature reduction on the knowledge base size. In the following, we consider two particular, more balanced requirement prioritizations. Analogously to minimal module extraction, we aim at preserving syntactic similarity of the extracted knowledge base, however based on the following, less restrictive similarity notions:

1. **Identical Sub-Expressions:**[2] Modules fulfill this notion of similarity, if they consist only of sub-expressions occurring in the original knowledge base.
2. **Structurally Equivalent Sub-Expressions**: Modules fulfill this notion of similarity, if for all of their sub-expressions there is a structurally equivalent sub-expression occurring in the original knowledge base, i.e. an expression with the same syntactic structure, but possibly different atomic concepts. For instance, $A \sqcap \exists r.A$ is structurally equivalent to $B_1 \sqcap \exists r.B_2$.

In the next sections, we investigate the task of knowledge base extraction based on these two notions of syntactic similarity with the second-highest priority given to the knowledge base size. We first show how we can extend the original knowledge base to contain all minimal general modules such that minimal module extraction can identify one of them. Subsequently, we add the computational complexity as a forth dimension. We investigate how we can obtain a tractable alternative to minimal module extraction by sacrificing the minimality guarantee, while fulfilling the requirement of syntactic similarity and reaching a decent effectiveness in terms of module size. In our evaluation, we show that, on average, the approach outperforms minimal module extraction applied directly to the original knowledge base.

4 Computing Modules Using Minimal Module Extraction

In this section, we show how, by normalizing the original knowledge base $\mathcal{T}$ and extending it with a subset of its deductive closure, we can obtain a union of all general modules fulfilling the syntactic similarity notions of identical sub-terms and structurally equivalent sub-terms. Applying minimal module extraction to this extended knowledge base would yield all minimal general modules, which can subsequently be ordered according to the signature size.

[2] Conjunctions containing a subset of conjuncts are not considered as sub-expressions.

In order to obtain a union of all minimal general modules under the restriction to identical sub-terms, we need to identify all subsumptions between sub-terms occurring in $\mathcal{T}$. We can structurally transform the knowledge base as follows: we assign a temporary concept name to each non-atomic sub-term occurring in $\mathcal{T}$, such that the knowledge base can be represented without nested expressions, i.e., using only axioms of the form $A \sqsubseteq B$, $A \equiv B_1 \sqcap \ldots \sqcap B_n$, and $A \equiv \exists r.B$, where A and $B_{(i)}$ are atomic concepts or $\top$. This can be realized in time linear in the size of $\mathcal{T}$ by recursively replacing complex concepts $C_{(i)}$ in expressions $C_1 \sqcap ... \sqcap C_n$ and $\exists r.C$ by fresh concept symbols with the corresponding equivalence axioms. Note that the original form of the knowledge base can easily be obtained by replacing the temporary concept names by their definitions.

By classifying the obtained knowledge base and extending it with the classification results, all subsumptions between sub-terms occurring in $\mathcal{T}$ are explicitly present in the resulting knowledge base $\mathcal{T}'$, which we call normalized $\mathcal{T}$. Thus, we obtain a polynomially-bounded, complete union of all possible general modules consisting of sub-terms occurring in $\mathcal{T}$ by replacing the temporary concept names by their definitions. Applying minimal module extraction to this knowledge base yields minimal general modules fulfilling the corresponding syntactic similarity requirement. In this way, we obtain a linear bound on the size of the general modules and, as demonstrated by Example 1, in most cases outperform minimal module extraction applied to the original knowledge base w.r.t. both, size of the signature and size of the general module.

If, in addition to identical sub-terms from $\mathcal{T}$, we can also use structurally equivalent sub-terms, we can introduce temporary concept names for all structurally equivalent sub-terms and then apply classification to obtain the corresponding dependencies. However, extending the knowledge base with all dependencies of the corresponding form and then applying minimal module extraction would lead to an increase of the overall complexity from exponential to double-exponential.

5 Computing Modules Using Rewriting

In the following, we propose a tractable approach to extracting general modules consisting of concepts structurally equivalent to sub-expressions of the original knowledge base. The approach is based on rewriting as it is used for uniform interpolation [7] and the aim of this section is to show how general modules are obtained using this rewriting.

In order to simplify the tracking of subsumption dependencies during the rewriting, we use the normalization introduced in the last section. Given a normalized $\mathcal{EL}$ knowledge base, the elimination of roles can be done by omitting all axioms with subsumees and subsumers containing irrelevant roles without loosing any relevant consequences. In the following, we focus, therefore, on the elimination of irrelevant concept names and assume w.l.o.g. that the sets of subsumees and subsumers do not contain any roles not from Σ.

During the rewriting, we keep two relations that map each atomic concept in a TBox to a set of concepts. These relations initially contain, for each atomic concept, the subsumees and subsumers as given by the normalized TBox. Each rewriting step then refines these relations in such a way that the union of all corresponding subsumption axioms is still a general module.

Definition 3. *Let $\mathcal{T}$ be a normalized $\mathcal{EL}$ knowledge base and $R^{\mathcal{T}}_{\sqsupseteq}, R^{\mathcal{T}}_{\sqsubseteq}$ relations that map each atomic concept $B \in sig_C(\mathcal{T})$ to a set of subsumees and a set of subsumers of B entailed by $\mathcal{T}$. Any pair $\langle R^{\mathcal{T}}_{\sqsupseteq}, R^{\mathcal{T}}_{\sqsubseteq}\rangle$ is called a* subsumee/subsumer relation pair for *$\mathcal{T}$ and it is called the* initial subsumee/subsumer relation pair for $\mathcal{T}$ *if $R^{\mathcal{T}}_{\sqsupseteq}$ and $R^{\mathcal{T}}_{\sqsubseteq}$ are as follows:*

1. $R^{\mathcal{T}}_{\sqsupseteq}(B) = \{C \mid C \sqsubseteq B \in \mathcal{T} \text{ or } C \equiv B \in \mathcal{T}\}$,
2. $R^{\mathcal{T}}_{\sqsubseteq}(B) = \{C \mid B \sqsubseteq C \in \mathcal{T} \text{ or } B \equiv C \in \mathcal{T}\}$.

If $\mathcal{T}$ is clear from the context, we simply write $\langle R_{\sqsupseteq}, R_{\sqsubseteq}\rangle$. Starting with the initial subsumee/subsumer relation, our rewriting aims at obtaining another pair of relations that allows for constructing a uniform interpolant as follows:

Definition 4. *Let $\langle R_{\sqsupseteq}, R_{\sqsubseteq}\rangle$ be a subsumee/subsumer relation pair and Σ a signature. We denote by $\Sigma^{\mathtt{ext}}(R_{\sqsupseteq}, R_{\sqsubseteq})$ the extension of Σ with atomic concepts occurring in the range of $R_{\sqsupseteq}$ and $R_{\sqsubseteq}$. We construct a knowledge base $\mathtt{M}(R_{\sqsupseteq}, R_{\sqsubseteq}, \Sigma)$ from $\langle R_{\sqsupseteq}, R_{\sqsubseteq}\rangle$ and Σ as:*

$$\begin{aligned}\mathtt{M}(R_{\sqsupseteq}, R_{\sqsubseteq}, \Sigma) = {} & \{C \sqsubseteq A \mid A \in \Sigma^{\mathtt{ext}}(R_{\sqsupseteq}, R_{\sqsubseteq}), C \in R_{\sqsupseteq}(A)\} \cup \\ & \{A \sqsubseteq D \mid A \in \Sigma^{\mathtt{ext}}(R_{\sqsupseteq}, R_{\sqsubseteq}), D \in R_{\sqsubseteq}(A)\} \cup \\ & \{C \sqsubseteq D \mid \textit{there is } A \notin \Sigma^{\mathtt{ext}}(R_{\sqsupseteq}, R_{\sqsubseteq}), C \in R_{\sqsupseteq}(A), D \in R_{\sqsubseteq}(A)\},\end{aligned}$$

If $\mathtt{M}(R_{\sqsupseteq}, R_{\sqsubseteq}, \Sigma) \in \mathtt{MOD}(\mathcal{T}, \Sigma)$, we say that $\langle R_{\sqsupseteq}, R_{\sqsubseteq}\rangle$ is complete *w.r.t. Σ.*

The above definition avoids an unnecessary extension of the knowledge base signature with atomic concepts in case $A \notin \Sigma^{\mathtt{ext}}(R_{\sqsupseteq}, R_{\sqsubseteq})$. Note that even in the initial subsumee/subsumer relation pair this case can occur, namely when concepts not from Σ do not have atomic subsumers or subsumees. We can show that the initial subsumee/subsumer relation pair meets the completeness criterion:

Theorem 1. *Let $\mathcal{T}$ be a normalized $\mathcal{EL}$ knowledge base, $\Sigma \subseteq sig(\mathcal{T})$ a signature, and $\langle R_{\sqsupseteq}, R_{\sqsubseteq}\rangle$ the initial subsumee/subsumer relation pair for $\mathcal{T}$, then $\langle R_{\sqsupseteq}, R_{\sqsubseteq}\rangle$ is complete w.r.t. Σ.*

Before defining the rewriting step that refines the initial subsumee/subsumer relation pair into another subsumee/subsumer relation pair preserving completeness, we show the initial subsumee/subsumer relation pair $\langle R_{\sqsupseteq}, R_{\sqsubseteq}\rangle$ and the according general module $\mathcal{T}_{\mathtt{M}} = \mathtt{M}(R_{\sqsupseteq}, R_{\sqsubseteq}, \Sigma^{\mathtt{ext}}(R_{\sqsupseteq}, R_{\sqsubseteq}))$ for the knowledge base $\mathcal{T}$ from Example 1. We first normalize $\mathcal{T}$ by introducing two temporary concepts, $B_1 \equiv A_{10} \sqcap A_{13}$ and $B_2 \equiv \exists r.A_9$, then we classify the normalized knowledge base and obtain the initial subsumee/subsumer relation pair shown in Fig. 1.

The knowledge base $\mathcal{T}_{\mathtt{M}}$ contains, for each A_i with $i \in \{1, \dots, 16\}$, all axioms of the form $C \sqsubseteq A_i$ with $C \in R_{\sqsupseteq}(A_i)$ and $A_i \sqsubseteq C$ with $C \in R_{\sqsubseteq}(A_i)$. This also holds for B_1 and B_2. It is not difficult to check that, after replacing B_1 by $A_{10} \sqcap A_{13}$ and B_2 by $\exists r.A_9$ in $\mathcal{T}_{\mathtt{M}}$, each axiom of $\{A_{12} \sqsubseteq A_{10}, A_{15} \sqsubseteq A_{13}, A_{10} \sqcap A_{13} \sqsubseteq A_{16}, A_{10} \sqsubseteq A_9, A_{13} \sqsubseteq A_9, A_9 \sqsubseteq \exists r.A_9, A_9 \sqsubseteq A_8, A_8 \sqsubseteq A_1\}$ is contained in it. Thus, the result contains, among other axioms, the general module given in Section 3 consisting of sub-expressions of $\mathcal{T}$, which shows completeness of $\langle R_{\sqsupseteq}, R_{\sqsubseteq}\rangle$.

	$R_{\sqsupseteq}$	$R_{\sqsubseteq}$
A_{16}	B_1	$\emptyset$
A_{15}	$\emptyset$	$A_1, \ldots, A_9, B_2, A_{13}, A_{14}$
A_{14}	A_{15}	$A_1, \ldots, A_9, B_2, A_{13}$
A_{13}	B_1, A_{14}, A_{15}	$A_1, \ldots, A_9, B_2$
A_{12}	$\emptyset$	$A_1, \ldots, A_{11}, B_2,$
A_{11}	A_{12}	$A_1, \ldots, A_{10}, B_2$
A_{10}	$B_1, A_{11}, A_{12}, A_{15}$	$A_1, \ldots, A_9, B_2$
A_9	$B_1, A_{10}, \ldots, A_{15}$	$A_1, \ldots, A_8, B_2$
A_8	$B_1, A_9, \ldots, A_{15}$	$A_1, \ldots, A_7$
A_7	$B_1, A_8, \ldots, A_{15}$	$A_1, \ldots, A_6$
A_6	$B_1, A_7, \ldots, A_{15}$	$A_1, \ldots, A_5$
A_5	$B_1, A_6, \ldots, A_{15}$	$A_1, \ldots, A_4$
A_4	$B_1, A_5, \ldots, A_{15}$	$A_1, \ldots, A_3$
A_3	$B_1, A_4, \ldots, A_{15}$	A_1, A_2
A_2	$B_1, A_3, \ldots, A_{15}$	A_1
A_1	$B_1, A_2, \ldots, A_{15}$	$\emptyset$
B_2	$\exists r.A_9, A_9, \ldots, A_{15}, B_1$	$\exists r.A_9$
B_1	$A_{10} \sqcap A_{13}$	$A_{10} \sqcap A_{13}, A_1, \ldots, A_{10}, A_{13}, B_2, A_{16}$

Fig. 1. The initial subsumee/subsumer relation pair $\langle R_{\sqsupseteq}, R_{\sqsubseteq}\rangle$ for Example 1

Since rewritings aiming at eliminating all irrelevant concept names yield smaller modules for sparse relation pairs, we will only use a subset of the subsumee/subsumer relations used as input for minimal module extraction. We compute a reduced subsumee/subsumer relation pair that only uses the transitive reduction of the classification results, i.e., we consider $B_1 \sqsubseteq B_2$ only if there is no B_3 such that $B_1 \sqsubseteq B_3$ and $B_3 \sqsubseteq B_2$. Furthermore, we compute, in polynomial time, a reduced graph by recursively eliminating subsumers and subsumees not from Σ that do not have any outgoing edges. It is easy to check that the completeness of the initial subsumee/subsumer relation pair stated in Theorem 1 still holds. In the next section, we assume this reduced form of initial subsumee/subsumer relation pair $\langle R_{\sqsupseteq}, R_{\sqsubseteq}\rangle$.

As demonstrated in Example 2, within the task of uniform interpolation, a single rewriting step replaces occurrences of an atomic concept in all subsumees and subsumers within a relation pair by its subsumees and subsumers, respectively. Since, in general, an atomic concept can have infinitely many subsumees and subsumers, using the whole set of subsumees and subsumers for rewriting is not feasible in practice. Interestingly, if the initial relation pair is complete, then a small subset of all subsumees and subsumers of the replaced concept is sufficient to preserve the completeness of the relation pair (in general, however, the sets of direct subsumees and subsumers are not sufficient). Among other things, the relevant subset does not need to include subsumees that can be obtained from other subsumees by adding arbitrary conjuncts to arbitrary sub-expressions. For instance, if B is a subsumee of A, then we do not need $B \sqcap B'$ for the replacement of A. Similarly, the minimal subset of subsumers required for replacement does not include concepts that can be obtained from other subsumers by omitting arbitrary conjuncts from arbitrary sub-expressions. While, in case of subsumees, a conjunction is not required if at least one of the conjuncts is a subsumee, in case of subsumers, we need to introduce a conjunction in particular when replacing an atomic concept within the scope of an existential restriction. Using the standard substitution notation $C[A/B]$ for denoting the concept obtained by replacing all occurrences of B within C by A, we give the following definition of an elementary rewriting.

Definition 5. *Let $\mathcal{T}$ be a normalized $\mathcal{EL}$ knowledge base, $\Sigma \subseteq sig(\mathcal{T})$ a signature, and $\langle R_{\sqsupseteq}, R_{\sqsubseteq}\rangle$ a subsumee/subsumer relation pair for $\mathcal{T}$. For atomic concepts $A, B \in sig_C(\mathcal{T})$ and $\bowtie \in \{\sqsupseteq, \sqsubseteq\}$, an* elementary rewriting $\mathtt{Rew}_{R_{\bowtie}}(B, C, A)$ *of a subsumee/subsumer $C \in R_{\bowtie}(B)$ w.r.t. A is given by*

1. $\mathtt{Rew}_{R_{\sqsupseteq}}(B, C, A) = \{(B, C') \mid A' \in R_{\sqsupseteq}(A), C' = C[A'/A]\}$.
2. $\mathtt{Rew}_{R_{\sqsubseteq}}(B, C, A) = \begin{cases} \{(B, C') \mid D' = \bigsqcap_{D \in R_{\sqsubseteq}(A)} D, C' = C[D'/A]\}, & (a) \\ \{(B, C') \mid A' \in R_{\sqsubseteq}(A), C' = C[A'/A]\}, & (b) \end{cases}$

where (a) is used when A is within the scope of an existential restriction and (b) is used otherwise. Let $S_A = \{(B, C) \mid C \in R_{\bowtie}(B)$ and A occurs in $C\}$. A rewriting w.r.t. A is given by $\mathtt{Rew}_{R_{\bowtie}}(A) = \bigcup_{(B,C)\in S_A} \mathtt{Rew}_{R_{\bowtie}}(B, C, A) \cup R_{\bowtie} \setminus S_A$.

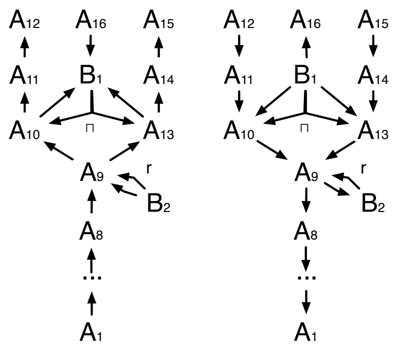
In order to keep the relations as small as possible, we further remove trivial subsumees and subsumers obtained during the rewriting, namely atomic concepts themselves and, in case of subsumee relations, conjunctions with the atomic concept itself as one of the conjuncts. This check is inexpensive from the computational point of view, since such trivial subsumees and subsumers can be identified independently from other subsumees and subsumers. In what follows, we assume that such trivial subsumees and subsumers are removed after each rewriting. We obtain the following result concerning the completeness w.r.t. Σ:

Theorem 2. *Let $\mathcal{T}$ be a normalized $\mathcal{EL}$ knowledge base, $\Sigma \subseteq sig(\mathcal{T})$ a signature, $\langle R_{\sqsupseteq}, R_{\sqsubseteq}\rangle$ a subsumee/subsumer relation pair for $\mathcal{T}$ that is complete w.r.t. Σ. Then, for any $B' \notin \Sigma$ holds $\langle \mathtt{Rew}_{R_{\sqsupseteq}}(B'), R_{\sqsubseteq}\rangle$ and $\langle R_{\sqsupseteq}, \mathtt{Rew}_{R_{\sqsubseteq}}(B')\rangle$ are subsumee/subsumer relation pairs for $\mathcal{T}$, which are complete w.r.t. Σ.*

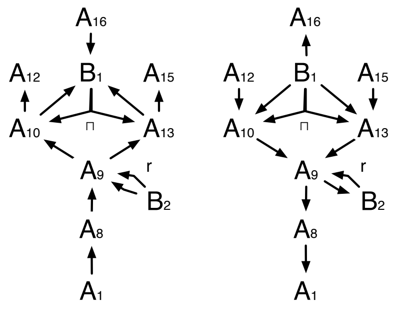
Thus, starting with the initial subsumee/subsumer relation pair $\langle R_{\sqsupseteq}, R_{\sqsubseteq}\rangle$, after each rewriting step we obtain a subsumee/subsumer relation pair over $\mathcal{T}$ that is complete w.r.t. Σ. However, without further restrictions, the above rewritings would potentially introduce many large nested concept expressions or might not even terminate. In the next section, we show how these problems can be avoided by stating the corresponding validity criteria for rewritings on subsumee/subsumer relation pairs.

6 Restricting Rewriting

In this section, we address the problems caused by unrestricted application of rewriting pointed out in Example 2. On the one hand, the example shows that rewriting can significantly change the syntactic structure of a knowledge base. On the other hand, it demonstrates that, while in some cases an elimination of a particular concept name can lead to a smaller knowledge base, it can cause the knowledge base to grow by several factors or even get infinite in other cases.



(a) Initial $R_{\sqsupseteq}$ (left) and $R_{\sqsubseteq}$ (right)



(b) Rewriting w.r.t. $A_2 - A_7$, A_{11}, A_{14}

Fig. 2. Hypergraphs for the knowledge base in Example 1

In order to avoid the above negative effects of rewriting, after each rewriting step we identify and exclude *invalid* rewritings, i.e., rewritings having a negative impact on the structure of the resulting module or the size of the relation pair. In particular, we exclude rewritings replacing atomic concepts by the conjunction of their direct subsumers corresponding to case (a) in Definition 5, since such a replacement possibly introduces concept expressions with a new structure not occurring in the original knowledge base. Thus, the set of valid rewritings is restricted to replacements of atomic concepts by their direct subsumees and subsumers. For the same reason, we additionally exclude rewritings yielding nested concept expressions, i.e., replacements of an atomic concept within a conjunction or existential restriction by one of its non-atomic subsumees or subsumers. Since the initial subsumee/subsumer relation pair contains only concepts of the form $B, \exists r.B$ and $B_1 \sqcap ... \sqcap B_n$, after each valid rewriting step, all subsumees and subsumers have also this simple form. In this way, subsumee/subsumer relations can be represented as hypergraphs with atomic concepts as nodes and three types of edges, namely $A \to B$ representing atomic subsumees/subsumers, $A \xrightarrow{r} B$ representing existential restrictions, and multi-edges $A \xrightarrow{\sqcap} B_1, ..., B_n$ representing conjunctions. The corresponding hypergraphs for the initial subsumee/subsumer relation pair $\langle R_{\sqsupseteq}, R_{\sqsubseteq}\rangle$ for the knowledge base in Example 1 are shown in Fig. 2(a).

In the following, we give a set of excluding conditions for rewritings according to the requirement of syntactic similarity and an inequation excluding rewritings negatively affecting knowledge base size. Within the excluding conditions, we distinguish three types of successors and predecessors according to the types of edges. For an atomic concept A and a relation $R_{\bowtie}$ with $\bowtie \in \{\sqsupseteq, \sqsubseteq\}$, we use

$$\begin{aligned}
\mathtt{IN}_A(A) &:= \{B \in \mathrm{sig}_C(\mathcal{T}) \mid A \in R_{\bowtie}(B)\}\\
\mathtt{OUT}_A(A) &:= R_{\bowtie}(A) \cap \mathrm{sig}_C(\mathcal{T})\\
\mathtt{IN}_{\mathtt{Roles}}(A) &:= \{B \mid \exists r.A \in R_{\bowtie}(B)\}\\
\mathtt{OUT}_{\mathtt{Roles}}(A) &:= \{B \mid \exists r.B \in R_{\bowtie}(A)\}\\
\mathtt{IN}_{\mathtt{Con}}(A) &:= \{B \mid B'_1 \sqcap ... \sqcap B'_n \in R_{\bowtie}(B) \text{ with } A = B'_i \text{ for some } i \in \{1, ..., n\}\}\\
\mathtt{OUT}_{\mathtt{Con}}(A) &:= \{B'_1 \sqcap ... \sqcap B'_n \mid B'_1 \sqcap \ldots \sqcap B'_n \in R_{\bowtie}(A)\}
\end{aligned}$$

Further, let $\mathtt{IN}(A) = \mathtt{IN}_A(A) \cup \mathtt{IN}_{\mathtt{Roles}}(A) \cup \mathtt{IN}_{\mathtt{Con}}(A)$ and $\mathtt{OUT}(A) = \mathtt{OUT}_A(A) \cup \mathtt{OUT}_{\mathtt{Roles}}(A) \cup \mathtt{OUT}_{\mathtt{Con}}(A)$.

In order to avoid an introduction of structurally new concept expressions during the rewriting and ensure termination, we exclude a rewriting w.r.t. an atomic concept A if one of the following conditions is true:

$$(\mathtt{IN}_{\mathtt{Roles}}(A) \cup \mathtt{IN}_{\mathtt{Con}}(A) \neq \emptyset) \text{ and } \mathtt{OUT}_A(A) \text{ contains temporary concepts;} \tag{1}$$

$$(\mathtt{IN}_{\mathtt{Roles}}(A) \cup \mathtt{IN}_{\mathtt{Con}}(A) \neq \emptyset) \text{ and } (\mathtt{OUT}_{\mathtt{Roles}}(A) \cup \mathtt{OUT}_{\mathtt{Con}}(A) \neq \emptyset); \tag{2}$$

$$R_{\bowtie} \text{ is a subsumer relation and } |\mathtt{IN}_{\mathtt{Roles}}(A)| \geq 1 \text{ and } |\mathtt{OUT}(A)| \geq 2; \tag{3}$$

$$\text{Some } C \in R_{\bowtie}(A) \text{ contains } A; \tag{4}$$

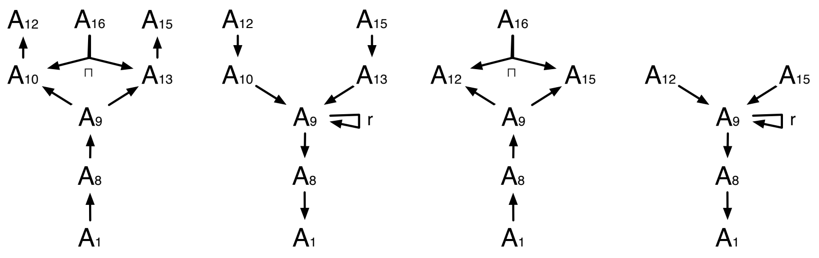
For instance, in Example 1 the rewriting w.r.t. A_9 in $R_{\sqsubseteq}$ is invalid due to Condition (3) and rewriting w.r.t. A_{10}, A_{13} in $R_{\sqsupseteq}$ are invalid due to Condition (2).

In order to identify rewritings that would increase the size of a relation, we compare the number of edges before and after the rewriting. While the number of edges potentially affected by a rewriting w.r.t. a concept A can be given by $|\mathtt{IN}(A)| + |\mathtt{OUT}(A)|$, the corresponding number of affected edges after the rewriting is in general bounded by $|\mathtt{OUT}(A)| + |\mathtt{IN}(A)| \cdot |\mathtt{OUT}(A)|$. Interestingly, if a concept B is unreferenced, it is usually possible to remove some elements from the corresponding sets $R_{\sqsupseteq}(B)$ and $R_{\sqsubseteq}(B)$ without losing any Σ consequences, or even without losing any axioms in $\mathtt{M}(R_{\sqsupseteq}, R_{\sqsubseteq}, \Sigma^{\mathtt{ext}}(R_{\sqsupseteq}, R_{\sqsubseteq}))$. We can remove subsumees and subsumers of unreferenced concepts, if none of the corresponding axioms in $\mathtt{M}(R_{\sqsupseteq}, R_{\sqsubseteq}, \Sigma^{\mathtt{ext}}(R_{\sqsupseteq}, R_{\sqsubseteq}))$ that contain these subsumees and subsumers, add any new Σ consequences to $\mathtt{M}(R_{\sqsupseteq}, R_{\sqsubseteq}, \Sigma^{\mathtt{ext}}(R_{\sqsupseteq}, R_{\sqsubseteq}))$. Thus, in order to determine if a subsumee $C \in R_{\sqsupseteq}(B)$ of $B \notin \Sigma$ is unnecessary, we check for each element $D \in R_{\sqsubseteq}(B)$, if $\mathtt{M}(R_{\sqsupseteq}, R_{\sqsubseteq}, \Sigma^{\mathtt{ext}}(R_{\sqsupseteq}, R_{\sqsubseteq})) \setminus \{C \sqsubseteq D\} \models C \sqsubseteq D$. Unnecessary subsumers can be determined in the same manner. For instance, in case of A_2 in Example 1, after the corresponding rewriting of both relations we can remove its subsumee A_3 and subsumer A_1, if $\mathtt{M}(R_{\sqsupseteq}, R_{\sqsubseteq}, \Sigma^{\mathtt{ext}}(R_{\sqsupseteq}, R_{\sqsubseteq})) \setminus \{A_1 \sqsubseteq A_3\} \models A_1 \sqsubseteq A_3$. It is easy to check given the corresponding hypergraphs that this is indeed the case. In fact, the corresponding sets of necessary subsumees and subsumers after the rewriting are empty for $A_2, \ldots, A_7$, A_{10}, A_{11}, A_{13}, A_{14} and B_1, B_2.

Given the relation $R_{\bowtie}^{\mathtt{red}}$ obtained by omitting such unnecessary elements from the set $R_{\bowtie}(B)$, we can use a tighter bound on the number of edges after rewriting based on $n_{R_{\bowtie}} = |R_{\bowtie}^{\mathtt{red}}(B)|$ instead of $|R_{\bowtie}(B)|$. Thus, we obtain the following inequation that holds for rewritings potentially increasing the size of relations:

$$|\mathtt{IN}(A)| + |\mathtt{OUT}(A)| < n_{R_{\bowtie}} + |\mathtt{IN}(A)| \cdot |\mathtt{OUT}(A)| \tag{5}$$

In Example 1, $|\mathtt{IN}_A(A_i)| = 1$ and $|\mathtt{OUT}_A(A_i)| = 1$ holds for all $i \in \{2, ..., 7, 11, 14\}$. Since both, $n_{R_{\sqsupseteq}}$ and $n_{R_{\sqsubseteq}}$ are 0 for all A_i, the number of edges decreases by one in case of each rewriting. After each rewriting including the subsequent omitting of unnecessary successors of the replaced concept, the number of edges as well as $n_{R_{\sqsupseteq}}$ and $n_{R_{\sqsubseteq}}$ remain the same for all remaining concepts. Thus, the conditions for the remaining concepts A_i with $i \in \{2, ..., 7, 11, 14\}$ do not change during any of the above rewritings. After performing all of the above rewritings, we obtain the subsumee/subsumer relation pair shown in Fig. 2(b).



(a) Rewriting w.r.t. B_1, B_2 (b) Rewriting w.r.t. A_{10}, A_{13}

Fig. 3. Rewriting for the knowledge base in Example 1

In case of B_1, in $R_{\sqsubseteq}$ we have only outgoing edges. Since both, $n_{R_{\sqsupseteq}}$ and $n_{R_{\sqsubseteq}}$ are 0, we can eliminate the concept from in $R_{\sqsubseteq}$ by omitting its subsumers. In $R_{\sqsupseteq}$, we have three incoming and one outgoing edge, i.e., Inequation (5) does not hold. The number of edges decreases also in this case, since two of the conjunction edges obtained by rewriting are trivial as specified in the last section and are removed directly after the rewriting. In case of B_2, we only need to consider $R_{\sqsubseteq}$, since in $R_{\sqsupseteq}$ the concept is already unreferenced. Since we again have one incoming and one outgoing edge and $n_{R_{\sqsubseteq}}$ is 0, we can also perform the corresponding rewriting and eliminate B_2, thereby obtaining the relation pair shown in Fig. 3(a).

Now, we can also perform rewriting w.r.t. A_{10}, A_{13} in $R_{\sqsupseteq}$, since Condition (2) does not hold any more. Checking for unnecessary subsumees and subsumers reveals that both, $n_{R_{\sqsupseteq}}$ and $n_{R_{\sqsubseteq}}$ are still 0 for both, A_{10} and A_{13}. Since Inequation (5) does not hold in any of the two graphs, we can perform the corresponding rewriting and eliminate both, A_{10} and A_{13}, thereby obtaining the relation pair shown in Fig. 3(b).

We recall that, in Example 1, $\Sigma = \{A_1, A_8, A_{12}, A_{15}, A_{16}, r\}$. Thus, the only atomic concept not from Σ still referenced within the subsumee/subsumer relations is A_9, which is not eligible for rewriting due to Condition (4). Therefore, the rewriting process is finished. After computing $\mathtt{M}(R_{\sqsupseteq}, R_{\sqsubseteq}, \Sigma^{\mathtt{ext}}(R_{\sqsupseteq}, R_{\sqsubseteq}))$, we obtain the smaller of the two general modules given in Example 1.

Algorithm 1 shows the rewriting process starting with the initial subsumee/subsumer relation pair $\langle R^0_{\sqsupseteq}, R^0_{\sqsubseteq}\rangle$. The computation terminates, when no further subsumees/subsumers could be eliminated during one iteration. We obtain a rewritten subsumee/subsumer relation pair $\langle R_{\sqsupseteq}, R_{\sqsubseteq}\rangle$ over $\mathcal{T}$ complete w.r.t. Σ, which is of a polynomial size in the size of the original (not normalized) knowledge base $\mathcal{T}_o$ and does not contain any nested concept expressions. Moreover, after replacing all temporary concept names in $\mathtt{M}(R_{\sqsupseteq}, R_{\sqsubseteq}, \Sigma^{\mathtt{ext}}(R_{\sqsupseteq}, R_{\sqsubseteq}))$ by their definitions, we obtain a general module of $\mathcal{T}_o$, which does not contain any structurally new concept expressions not occurring in $\mathcal{T}_o$. We can summarize the results as follows.

Theorem 3. *Let $\mathcal{T}$ be a normalization of an $\mathcal{EL}$ knowledge base $\mathcal{T}_o$, $\Sigma \subseteq sig(\mathcal{T}_o)$ a signature, $\langle R_{\sqsupseteq}, R_{\sqsubseteq}\rangle$ the output of Algorithm 1 for an initial subsumee/subsumer pair $\langle R^0_{\sqsupseteq}, R^0_{\sqsubseteq}\rangle$ of $\mathcal{T}$. For $\mathcal{T}_r$ the knowledge base obtained by replacing all temporary concept names in $\mathtt{M}(R_{\sqsupseteq}, R_{\sqsubseteq}, \Sigma^{\mathtt{ext}}(R_{\sqsupseteq}, R_{\sqsubseteq}))$ by their definitions:*

Algorithm 1. Rewriting of Subsumee/Subsumer Relation Pairs

```
Data: $\langle R^0_\sqsupseteq, R^0_\sqsubseteq \rangle$ initial subsumee/subsumer relation pair for a normalized knowledge
      base
Result: $\langle R_\sqsupseteq, R_\sqsubseteq \rangle$ rewritten subsumee/subsumer relation pair
1  $\langle R_\sqsupseteq, R_\sqsubseteq \rangle \leftarrow \langle R^0_\sqsupseteq, R^0_\sqsubseteq \rangle$;
2  while fixpoint is not reached do
3      for $B \in sig_C(\mathcal{T}) \setminus \Sigma$ do
4          if Conditions (1)–(4) are false then
5              $n_{R_\sqsupseteq} \leftarrow |R^{\mathtt{red}}_\sqsupseteq(B)|$;
6              $n_{R_\sqsubseteq} \leftarrow |R^{\mathtt{red}}_\sqsubseteq(B)|$;
7              if Inequation (5) does not hold then
8                  $R_\sqsupseteq \leftarrow \mathtt{Rew}_{R_\sqsupseteq}(B) \setminus (R_\sqsupseteq(B) \setminus R^{\mathtt{red}}_\sqsupseteq(B))$;
9                  $R_\sqsubseteq \leftarrow \mathtt{Rew}_{R_\sqsubseteq}(B) \setminus (R_\sqsubseteq(B) \setminus R^{\mathtt{red}}_\sqsubseteq(B))$;

10 return $\langle R_\sqsupseteq, R_\sqsubseteq \rangle$;
```

– $\mathtt{M}(R_\sqsupseteq, R_\sqsubseteq, \Sigma^{\mathtt{ext}}(R_\sqsupseteq, R_\sqsubseteq))$ *can be computed in polynomial time and is polynomial in the size of* $\mathcal{T}_o$*;*
– *for all sub-expressions* C' *occurring in* $\mathcal{T}_r$ *there is a sub-expression* C *of* $\mathcal{T}_o$ *such that* C' *can be obtained from* C *by exchanging atomic concepts.*

7 Evaluation

For our evaluation, we use the $\mathcal{EL}$ fragment of the Gene Ontology[3] describing gene product characteristics in terms of how gene products behave in a cellular context. The OWL version of the ontology (April 2012) comprises 36,251 atomic classes, 8 object properties and 316,580 logical axioms, out of which 66,117 axioms are terminological (the $\mathcal{EL}$ fragment contains 66,101 terminological axioms).

We implemented our approach in Java based on the OWL-API. The aim of the evaluation is to compare the results of our approach in terms of module size and computation time to minimal module extraction and *Locality-based extractor* [1] – an existing tractable approach to (not necessarily minimal) module extraction not based on rewriting. To the best of our knowledge, there are currently no existing implementations of minimal module extraction for $\mathcal{EL}$, but only for DL-Lite$_{\mathtt{bool}}$ [3]. Therefore, we compare the two implementations on the DL-Lite$_{\mathtt{bool}}$ fragment of $\mathcal{EL}$, obtained from an $\mathcal{EL}$ knowledge base by replacing qualified existential restrictions by the corresponding unqualified restrictions. In order to also estimate the difference in the module size for $\mathcal{EL}$, we implemented a module extractor based on minimal justifications, which, given a general module obtained using our approach, computes a subset of the original ontology entailing the general module.

For the evaluation, we use signatures with 10, 30 and 50 atomic concepts and 4 roles each. For each signature size, we randomly choose 10 signatures and let the

[3] http://www.geneontology.org/

Table 1. Evaluation results on the DL-Lite$_{\texttt{bool}}$ fragment of $\mathcal{EL}$

Signature size	Rewriter	Minimal module extractor	Locality-based extractor
10	4.8	9.7 (2.0)	167 (34.8)
30	10.3	22.2 (2.2)	436 (41.1)
50	28.8	60.4 (2.1)	1245 (43.2)

Table 2. Evaluation results on $\mathcal{EL}$

Signature size	Rewriter	Minimal justification extractor	Locality-based extractor
10	21	43 (2.0)	259 (12.3)
30	45	104 (2.3)	659 (14.6)
50	151	306 (2.0)	1787 (11.8)

different extractors compute the corresponding general module. Subsequently, we compute the average module size, shown in Tables 1 and 2 (the number in brackets is the average module size measured in the corresponding average size of the modules computed by the rewriter). The first table shows the results for the DL-Lite$_{\texttt{bool}}$ fragment of $\mathcal{EL}$. Due to the lower expressivity, the obtained DL-Lite$_{\texttt{bool}}$ modules are considerably smaller than their $\mathcal{EL}$ correspondents in Table 2. We observe that the size of the minimal DL-Lite$_{\texttt{bool}}$ modules containing only axioms from the original knowledge base $\mathcal{T}$ are between 2.0 and 2.2 times larger than the corresponding general modules consisting of sub-expressions of $\mathcal{T}$ with possibly exchanged atomic concepts obtained using rewriter. The corresponding DL-Lite$_{\texttt{bool}}$ modules obtained by the locality-based extractor are even between 34.8 and 43.2 times larger. In case of $\mathcal{EL}$ modules, the minimal justifications of the general modules computed by rewriter are between 2.0 and 2.3 times larger, while the modules obtained by the locality-based extractor are between 11.8 and 14.6 times larger.

We further analyzed whether different proportions of particular axiom types influence the effectiveness of rewriter, but did not find this to be the case. Our conjecture is that this is a result of the simple structure of GO, which contains only axioms referencing exactly two atomic concepts, e.g., atomic subsumptions and existential restrictions. In case of such axioms, exactly two substitutions of atomic concepts are possible, each of which can potentially replace a justification consisting of several axioms, e.g., a chain of subsumption axioms. Since the effect of each such replacement is not dependent on the axiom type, but rather on the referenced concepts, this conjecture seems reasonable.

Concerning the computation time, we observe a significant difference between the tractable approaches (rewriter and the locality-based extractor) and the minimal module extractor. While, for the signatures with 50 atomic concepts, the first two approaches require less than one minute, minimal module extractor required between two hours and two days depending on the signature.

8 Summary

In this paper, we show that knowledge base extraction gains in effectiveness in terms of knowledge base size, when modules are not required to be subsets of the original knowledge base. We investigate the task of knowledge base extraction for $\mathcal{EL}$ based on two alternative, less restrictive notions for syntactic similarity.

First, we discuss the extraction of knowledge bases consisting only of sub-expressions occurring in the original knowledge base. We show how minimal modules fulfilling this similarity requirement can be obtained in ExpTime by introducing temporary concept names for complex concepts, adding a subset of the deductive closure to the knowledge base and subsequently applying minimal module extraction.

Second, we consider the extraction of modules that consist of concepts structurally equivalent to sub-expressions occurring in the original knowledge base.We propose a tractable approach that, in most cases, yields small knowledge bases, but does not guarantee the minimality of the result. As we show in our evaluation, modules extracted during our evaluation using minimal module extractor for DL-Lite$_{\texttt{bool}}$ are 2.0 to 2.2 times larger than those obtained by our approach. In case of $\mathcal{EL}$, knowledge bases obtained by our rewriter on average contain half as many axioms as their minimal justifications within the original knowledge base. In case of the locality-based module extractor, the extracted $\mathcal{EL}$ modules are on average 12 times larger than the general modules obtained by our approach.

Acknowledgements. This work is supported by the German Federal Ministry of Education and Research (BMBF) under grant 02PJ1002 (SyncTech).

References

1. Cuenca Grau, B., Horrocks, I., Kazakov, Y., Sattler, U.: Just the right amount: extracting modules from ontologies. In: Proc. of the 16th Int. Conf. on World Wide Web (WWW 2007), pp. 717–726 (2007)
2. Konev, B., Walther, D., Wolter, F.: Forgetting and uniform interpolation in large-scale description logic terminologies. In: Proc. of the 21st Int. Joint Conf. on Artificial Intelligence (IJCAI 2009), pp. 830–835 (2009)
3. Kontchakov, R., Wolter, F., Zakharyaschev, M.: Logic-based ontology comparison and module extraction, with an application to DL-Lite. Artificial Intelligence 174, 1093–1141 (2010)
4. Lutz, C., Wolter, F.: Foundations for uniform interpolation and forgetting in expressive description logics. In: Proc. of the 22nd Int. Joint Conf. on Artificial Intelligence (IJCAI 2011), pp. 989–995 (2011)
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. Nikitina, N., Glimm, B.: Hitting the sweetspot: Economic rewriting of knowledge bases. Techreport, AIFB, KIT, Karlsruhe (May 2012)
7. Nikitina, N., Rudolph, S.: ExpExpExplosion: Uniform interpolation in general EL terminologies. In: Proc. of the 20th European Conf. on Artificial Intelligence, ECAI 2012 (2012)
8. Nikitina, N., Rudolph, S., Glimm, B.: Reasoning-supported interactive revision of knowledge bases. In: Proc. of the 22nd Int. Joint Conf. on Artificial Intelligence (IJCAI 2011), pp. 1027–1032 (2011)
9. OWL Working Group, W.: OWL 2 Web Ontology Language: Document Overview. W3C Recommendation (October 27, 2009), http://www.w3.org/TR/owl2-overview/

Mining Semantic Relations between Research Areas

Francesco Osborne[1] and Enrico Motta[2]

[1] Dept. of Computer Science, University of Torino, 10149 Torino, Italy
osborne@di.unito.it
[2] Knowledge Media Institute, The Open University, MK7 6AA, Milton Keynes, UK
e.motta@open.ac.uk

Abstract. For a number of years now we have seen the emergence of repositories of research data specified using OWL/RDF as representation languages, and conceptualized according to a variety of ontologies. This class of solutions promises both to facilitate the integration of research data with other relevant sources of information and also to support more intelligent forms of querying and exploration. However, an issue which has only been partially addressed is that of generating and characterizing semantically the relations that exist between research areas. This problem has been traditionally addressed by manually creating taxonomies, such as the ACM classification of research topics. However, this manual approach is inadequate for a number of reasons: these taxonomies are very coarse-grained and they do not cater for the fine-grained research topics, which define the level at which typically researchers (and even more so, PhD students) operate. Moreover, they evolve slowly, and therefore they tend not to cover the most recent research trends. In addition, as we move towards a semantic characterization of these relations, there is arguably a need for a more sophisticated characterization than a homogeneous taxonomy, to reflect the different ways in which research areas can be related. In this paper we propose Klink, a new approach to i) automatically generating relations between research areas and ii) populating a bibliographic ontology, which combines both machine learning methods and external knowledge, which is drawn from a number of resources, including Google Scholar and Wikipedia. We have tested a number of alternative algorithms and our evaluation shows that a method relying on both external knowledge and the ability to detect temporal relations between research areas performs best with respect to a manually constructed standard.

Keywords: Research Data, Ontology Population, Bibliographic Data, Empirical Evaluation, Scholarly Ontologies, Data Mining.

1 Introduction

Consistently with the general trend towards characterizing information using Semantic Web standards, for a number of years now we have seen the emergence of repositories of research outputs specified using OWL/RDF as representation languages – e.g., see [1], [2], [3], and conceptualized according to a variety of ontologies, such as SWRC[1], BIBO[2], and AKT[3]. This class of solutions promises both to facilitate the integration of research data with other relevant sources of information – e.g., data drawn from social media [4], and also to support more intelligent forms of querying and exploration. In particular, in order to make sense of the key trends and dynamics of a research area, it is essential to have tools which are able to support a seamless exploration of the various relations that exist between authors, publications, impact measures, publication venues, research areas, etc. Within this context, it is particularly important to associate correctly authors and publications to research areas, to ensure good precision and recall when exploring what goes on within a particular research area.

The association between authors and publications on the one hand and research areas on the other is normally determined on the basis of the keywords that authors themselves associate with their publications. However, this purely syntactic approach is unsatisfactory for a number of reasons: authors do not necessarily use a consistent terminology to specify the relevant research areas and, even when they do, a syntactic approach fails to capture the relations that may exist between research areas – e.g., most researchers consider “ontology alignment” and “ontology matching” as essentially equivalent labels for the same research area, but searching for “ontology alignment” in most bibliographic servers does not return papers tagged as “ontology matching”. Hence, there is a need for methods which are able to generate the relations which exist between research areas, to enable more intelligent querying and exploration of research data.

This problem has been traditionally addressed by manually creating taxonomies, such as the ACM classification[4]. However, this manual approach suffers from a number of problems. These taxonomies are very coarse-grained and they do not cater for the fine-grained research topics, which define the level at which typically researchers (and even more so, PhD students) operate. Moreover, because these taxonomies are defined manually, they evolve slowly, and therefore they do not cover the most recent research trends. In addition, as we move towards semantically characterized repositories of research data, there is arguably a need for a more sophisticated representation of the relations between research areas, than a homogeneous and un-typed taxonomy, to reflect the different ways in which research areas can be related.

In this paper we address this problem by proposing Klink, a new approach to automatically generating relations between research areas, which combines both machine learning methods and external knowledge, drawn from a number of resources, including Google Scholar and Wikipedia. In particular, we have tested a number of alternative algorithms and our evaluation shows that a method relying on both external knowledge and temporal relations between research areas performs best with respect to a manually constructed standard and indeed achieves a very good level of precision and recall.

[1] `http://ontoware.org/swrc/`

[2] `http://bibliontology.com`

[3] `http://www.aktors.org/publications/ontology`

[4] `http://www.acm.org/about/class/ccs98-html`

2 What's in a Link: Characterizing Relations between Research Areas

Taxonomies of research areas are not like taxonomies in other domains, in the sense that there is not necessarily an all-encompassing and 'objective' organization of research topics. For example, one of the authors of this paper was involved in one of the very first attempts at building a semantic repository of research data, the KA2 initiative [5], and participated in a workshop whose main goal was to generate a taxonomy of research topics. This turned out to be much harder than predicted, given that for a number of topics there were serious disagreements about their relationships with other topics. Nevertheless, it is also the case that, given a research community, there are typical many relatively unproblematic cases where a broad consensus can be found about an area being equivalent to or being a sub-area of another area. For instance, we earlier made the example of the terms "ontology alignment" and "ontology matching" being used practically as synonyms in the research community. Another relatively uncontroversial example concerns the area of Semantic Web Services, which most people agree is a sub-area of both Web Services and Semantic Web.

However there are also other situations that are rather less obvious. For instance, while there may be a certain degree of consensus that research in Ontology Engineering is relevant to the Semantic Web area, most people would disagree with the statement that Ontology Engineering is a sub-area of Semantic Web. Nevertheless, if I am looking for papers on the Semantic Web, it may actually be useful for me if my system for research data exploration were also able to flag papers in Ontology Engineering as potentially relevant. And indeed, the relevance of the latter area of research to the former can be easily ascertained by browsing the proceedings of the main Semantic Web conferences.

In sum, the point here is that simply looking either for strict equivalence between research areas or strict *subAreaOf* relations is unsatisfactory, because it may fail to capture some other useful relations between research areas. For this reason, in our work so far we have also included relations such as that exemplified by Ontology Engineering and Semantic Web, where the results from the former contribute to research in the latter. Hence, our model at the moment considers the following three relations between research areas:

- *relatedEquivalent*. This is defined as a sub-property of *skos:related*, which indicates that two particular ways of referring to research areas can be treated as equivalent for the purpose of exploring research data – e.g., "ontology alignment" and "ontology matching" can be considered as equivalent.
- *skos:broaderGeneric*. We reuse this property from the SKOS[5] model, to indicate that a research area – e.g., Web Services, is broader than Semantic Web Services. Transitivity is important here, because this property is used to characterize the intuitive notion that an area is a sub-area of another one.

[5] `http://www.w3.org/2004/02/skos/`

- *contributesTo*. This is defined as a sub-property of *skos:related* and indicates that while an area, R1, is not a sub-area of another one, R2, its research outputs are so relevant to R2 that it may be useful for the purposes of querying and exploration to assert this relationship, to provide better support to users.

However, it is important to emphasize that, while our epistemology distinguishes between the aforementioned three relations, the current version of our algorithm, which will be presented in the next section, is only able to differentiate automatically between *hierarchical* and equivalent relations. In other words, while the algorithm is able to differentiate *relatedEquivalent* relations from the others, and it is also able to mine both *contributesTo* and *skos:broaderGeneric* relations, it treats these two relations as generic hierarchical relations and cannot differentiate them further. Hence, this final step – i.e., separating *contributesTo* from *skos:broaderGeneric* relations, needs at the moment to be carried out manually.

Our model[6] builds on the BIBO ontology, which in turn builds on SKOS[7], FOAF[8], and other standards. Our goal here was not to produce yet another ontology, so our extensions to BIBO are very conservative and comprise only the *relatedEquivalent* and *contributesTo* object properties described earlier, and the class *Topic*, which is used to refer to research topics.

3 The Klink Algorithm: Automatically Detecting Relations between Research Areas

3.1 Preliminaries

We propose a novel approach, named Klink, for cleaning and inferring hierarchical and equivalence relationships from a set of keywords associated with a collection of documents.

Klink detect links between keywords by using heuristic rules, statistical methods and external knowledge. Moreover it allows a human user to define some aspects of the hierarchy, such as the maximum permitted number of parent nodes for each node.

---

[6] `http://kmi.open.ac.uk/technologies/rexplore/ontologies/BiboExtension.owl`

[7] The most recent specification of the SKOS model, which can be found at http://www.w3.org/TR/2009/REC-skos-reference-20090818/, makes a number of modifications to the modeling of these relations and in particular proposes a new property, *skos:broaderTransitive,* to support the representation of transitive hierarchical relations. Here we stick to the older SKOS specification, primarily because our conceptual model builds on the BIBO ontology, which in turn builds on the 2004 SKOS model. While there are interesting semantic differences between the different versions of the SKOS model, in the context of this paper these are not so important, as we are only concerned with extracting the three kinds of relations between research areas, which have been presented above.

[8] `http://xmlns.com/foaf/spec/`

An important aspect of Klink is that it is able to discard keywords which are not research areas but can be used as keywords for a paper. Typical examples include names of software tools as well as ‘orthogonal’ keywords, e.g., “Case Study”, which do not denote a research area but a particular aspect of the paper in question – i.e., that a case study is presented.

Since we use a statistical approach it is imperative to have an unbiased and large enough collection of documents. To do any kind of inference on a keyword that has a low number of occurrences may be risky; it is better to discard it, at the cost of losing some useful piece of information. We should also be careful not to introduce biases when extracting subsets from a larger population. For example if we were to analyze a sample composed only by papers from the five best Semantic Web conferences, the importance of Semantic Web with respect to other areas would be necessarily overestimated. In that sample we may in fact discover that 80% of the papers about Machine Learning are associated with Semantic Web, and thus erroneously conclude that Machine Learning is a sub-area of Semantic Web. For this reason, while our experiments zoom on the Semantic Web as the ‘focus topic’, the corpus we use is very large and includes more than one million and half papers downloaded from Microsoft Academic Search[9] (MAS), which by and large are situated in the Computer Science area.

3.2 Overview of the Approach

The input to Klink is a collection of keywords associated with a set of documents and the result is a graph structure containing both hierarchical and equivalence links. The outline of the algorithm is as follows:

1) Each keyword in input is compared to the other keywords with which it shares at least *n* co-occurrences and two kinds of hierarchical links are inferred: the *‘standard’* one and the *‘temporal’* one;
2) Each keyword is checked for possible deletion if it does not meet the requirements for being a research area;
3) The links are cleaned by deleting triangular and circular hierarchical relationships and the eventual user’s requirements on the structure are enforced;
4) Each keyword is compared to the other keywords with which it shares at least *n* co-occurrences; the *relatedEquivalent* relationships are inferred and the relative keywords are merged;
5) Step 1, 3 and 4 are repeated with the new keywords obtained by merging the keywords with inferred equivalence relationships until no new *relatedEquivalent* relationships emerge.

It should be noticed that step 2 will be run only once and, as a choice, can be applied after step 5, giving the keywords that should be deleted the possibility of entering into a *relatedEquivalent* relationship.

[9] `http://academic.research.microsoft.com`

3.3 Step 1 – Inferring Hierarchical Relationships

In the classical definition of subsumption [6], term x is said to subsume term y if two conditions hold: $P(x|y) = 1$ and $P(y|x) < 1$, e.g. if y is associated to documents that are a subset of the documents x is associated to. Usually the first condition is relaxed in $P(x|y) > \alpha$, since it is quite improbable to find a perfect relationship. The usual value of α is 0.8, although other values are possible according to the kind of documents examined. For the inference of a hierarchical relationship between keywords we use a variation of this idea, combined with other heuristic metrics. We consider two different kinds of links, the standard one and the temporal one.

3.3.1 Inferring Standard Hierarchical Links

We define as a hierarchical link of x with respect to y the relationship in which the difference between $P(y|x)$ and $P(x|y)$ leans decidedly toward y and the two terms co-occur with a similar set of keywords.

We compute the strength of the hierarchical relationship as:

$$L(x,y) = (P(y|x) - P(x|y)) * c(x,y) * (1 + N(x,y))$$

where $c(x, y)$ is the cosine similarity between keywords and $N(x,y)$ is a metric that weighs the similarity of the keyword names. This similarity is computed as the ratio between the number of identical words between two keywords and their average number of words.

A hierarchical link is inferred when $L(x,y) > t$, and thus x is considered a sub-area of y. We suggest a value of 0.2 for t, and in the evaluation we will show how recall and precision change for different values of t. It is also possible to use other additional filters, chosen carefully according to the set of documents. We actually experimented with some of them on the sample of metadata downloaded from MAS, obtaining interesting results. Specifically we used the condition that a keyword had to be at least two years older than another as a necessary (but not sufficient) condition for being considered as its super-area. We then experimented with a filter based on dimensions, accepting as sub-areas only areas n times smaller then the super-area. However this technique can bring more problems than advantages, since an area can outgrow its super-area. Our conclusion was that this filtering technique might be useful but it is strongly dependent on the characteristics of the selected collection of documents.

3.3.2 Inferring Temporal Hierarchical Links

Some relationships among areas may escape the mechanism previously described. Usually, when an area is mature enough, references to its super-area become implicit and no longer appear as co-occurring keywords. For example many disciplines fall under the Artificial Intelligence area, but today it is hard to find explicit references to it in the keywords associated with a publication. For a human it is not very informative to annotate that Machine Learning is a form of Artificial Intelligence, since it is already a huge and independent research area by itself. The information about the origins of a research area is however necessary when building a complete taxonomy.

As an area grows, the co-occurrences with its super-area become fewer and fewer, making harder to infer the origins of a topic by looking only at the total co-occurrences. Taking into consideration also the temporal dimension aims to solve this drawback. The idea is that the initial co-occurrences of two keywords are the most informative about stating that a parent area somehow spawned a sub-area.

We use the term *temporal link* to refer to the relationship behind this intuition. It should be noted that, although temporal links can be used together with the standard links to build a taxonomy, they have a different meaning. A standard link between *x* and *y* implies that *y* was a vital keyword for *x* along the total life of *x*. The temporal link instead implies that the set of keywords with which *x* co-occurred in its first years are privileged. Therefore a temporal link between *x* and *y* means that *y* was a very important keyword for *x* in the initial years of *x*'s life. As time goes by, the relationship with the topic inferred by means of the temporal link may persist, or become implicit or even vanish.

The temporal weighted co-occurrence $CO_t(x,y)$ is obtained by adding over the years the number of co-occurrences weighted by a factor $w(year,x)$ given by:

$$w(year,x) = (debut(x) - year)^{-\gamma}$$

where $debut(x)$ is the year of the debut of keyword *x*, *year* is the year in which a co-occurrence occurs, and γ is a constant > 0 that modules the importance of co-occurring in a given year. We empirically set this value to 2. It is advisable not to use as debut the first year in which a keyword appeared, but rather the first year in which it appeared in at least a minimum number of papers. We use as limit 30 papers, but according to our tests any number between 10 and 50 gives reasonable results. To reduce the noise it is also possible to take in consideration only the first *n* years.

The temporal subsumption metrics $L_t(x,y)$ is computed as for the standard link, using the temporal conditional probability $P_t(x|y)= CO_t(y,x)/ CO_t(y,y)$:

$$L_t(x,y) = (P_t(y|x) - P_t(x|y)) * c(x,y) * (1+N(x,y))$$

As before, a temporal link is inferred if $L_t(x,y)>t_t$. As for the standard link, we suggest a threshold value of 0.2, and in the evaluation we will show the outcome for different values of t_t.

3.3.3 Integrating External Knowledge

Using external knowledge to integrate the examined corpus of documents is not always necessary but it can be very useful.

We often want to build a general taxonomy that reflects more the common use of a keyword in a certain domain rather than its interpretation in a particular set of documents. The examined set may in fact use keywords with a non-common meaning or be biased in some way. In this case it is advisable to rely on a neutral source of information [7]. Moreover, as it will be shown in section 3.4, external knowledge is vital to discover and discard keywords that do not belong to a certain domain.

We focused on the knowledge of the dimension of a keyword and of the co-occurrences of keyword pairs in different online sources. By choosing the right sources we can be sure to obtain this knowledge within a given domain – e.g., the academic one. We used parsers to collect this kind of knowledge from Google[10], Google Scholar[11], Wikipedia[12], and Eventseer[13].

Google and Wikipedia give a good approximation of keyword presence in general. On the contrary, Google Scholar focuses on the academic domain and in particular on the title and abstract of papers. Eventseer is a site that collects calls for papers, and as a result is very useful for understanding the dynamics among keywords as conferences topics. We used Google to search in both Wikipedia and Eventseer.net since the internal search of these services do not support the AND conjunction. We then exploited the "About […] results" text for estimating the occurrence of a keyword. For the co-occurrence we used the same technique, using the AND between them in the query. We also experimented with the OR conjunction, but the combination of AND with OR seemed to yield inconsistent results.

We computed the external probability as the weighted average of the probability of a co-occurrence for each different source: $P_{ex}(x|y) = \Sigma_i w_i P_i(x|y)$. In the evaluation we considered Wikipedia (w=0.2), GoogleScholar (w=0.4) and Eventseer (w=0.4) and we computed the hybrid probability as

$$P_h(x|y)= w_h P(x|y) + (1-w_h)P_{ex}(x|y)$$

where $0<w_h<1$ is the constant that reflects the importance of the external probability. We set this value to 0.5 to balance the contributions of the two components. When the set of document is not very large it may instead make sense to rely more heavily on external knowledge. We can then compute the hybrid version of $L(x,y)$ by simply using the hybrid probability $P_h(x|y)$ instead of the standard one.

It is important to mention that it is not possible to use external knowledge from the aforementioned sources to deduce temporal links, given that these sources do not provide the distribution of the co-occurrences over the years.

3.4 Step 2 – Cleaning the Keywords

When creating a taxonomy it is important to identify the keywords that are part of a certain domain, in this case the domain of 'research areas', and those that are not.

Text mining techniques may lead to noisy keywords that do not add any information and actually risk spoiling the inference process. For example in the MAS dataset we can find different keywords that are related to the academic world but is difficult to consider as research areas – e.g., "Web Pages", "Case Study", "Java Applet", and others. Hence it is important to detect and discard them from the final taxonomy. Our approach implements three techniques to filter this kind of irrelevant keywords.

The first and simplest procedure is the elimination of any keyword without inferred relationships with other keywords.

[10] www.google.com
[11] scholar.google.com
[12] www.wikipedia.org
[13] eventseer.net

The second technique uses the distribution of the keyword co-occurrences. An acceptable keyword should have a limited set of main keywords with which it has a relatively high number of co-occurrences and then a long tail of less important one. Some keywords show instead a flatter distribution of co-occurrences over a large range of keywords. This is the case of many general words used in the academic world, such as "Case Study", which can occur in many papers on completely different topics. We identify these spurious keywords by fixing the number of main keywords and the minimum percentage of co-occurrences they should cover. If the main keyword covers too small a part of the total co-occurrences, then the keyword is discarded as being too general. In the evaluation we used as thresholds 20 and 15%, respectively.

The third technique uses external knowledge and it is basically a check on the estimated dimension of a keyword in a certain domain. To do so we compute the weighted sum of the ratios between the dimension of a given keyword and the average dimension of the keywords in the various sources:

$$D_{ex}(x)= \Sigma_i\, w_i\, (D_i(x) / A_i)$$

where A_i and w_i are respectively the average dimension and the relative importance of the keyword in the i-th dataset of the specific source.

Google and Wikipedia are less useful sources to consider when we want to know the dimension of a keyword in the academic world. Hence, we give more importance to conference calls (Eventseer) than to the occurrences in the title or in the abstract of a paper (GoogleScholar), by setting w_{ev}=0.6 and w_{gs}=0.4. If $D_{ex}(x)$ is below a given threshold, which we set empirically at 0.2, the keyword is dropped.

There may be keywords that have a small dimension but are nevertheless real research areas. Thus before deleting a keyword we run a check on its links: if either a normal or temporal link has a strength that is at least the double of the correspondent threshold, then the keyword is kept.

3.5 Step 3 – Cleaning the Links

After step 2 we have a large number of cases in which two super-areas of a keyword are also in a hierarchical relationship. For example Word Wide Web and Semantic Web may both be super nodes of OWL whereas Word Wide Web may be also a super node of Semantic Web. Since such a taxonomy might be confusing the redundant links like the one between Word Wide Web and OWL are deleted.

In this stage, it is possible to cut away the links with lower $L(x,y,)$ or $L_t(x,y)$ to satisfy the user's requirements on the maximum number of super and sub nodes.

3.6 Step 4 and 5 – Detection of *relatedEquivalent* Relationships and Merging of the Keywords

The search for *relatedEquivalent* relationships between keywords offers many advantages. For example we can learn that "P2P" and "Peer to Peer" are actually the same topic and thus a query for any of the two will return a set of documents associated with both these keywords. Any statistical inference on the Peer to Peer area can then use a larger number of papers, and thus be more valid. *relatedEquivalent* relationships can be very important also when focusing only on building a taxonomy since they simplify the structure, making the subsumption inference easier.

A standard metric like the cosine similarity may work well in some cases but it raises two problems. The first is due to the fact that the eventual subsumption relationship between the keywords is not considered. If one of the keyword subsumes in some sense the other, a hierarchical link is preferable to a *relatedEquivalent* relationship. The second problem is that it is important to take in account the reasons why two keywords have a high cosine distance. In a taxonomy it is normal for sibling elements in the lower levels to have a high cosine similarity since they are different declinations of the same theme. Thus we need to take in consideration also the cosine similarity of the common super-areas of these keywords, namely the keywords that subsume both of them. If the cosine similarities of the two keywords with the common super-areas are comparable with their reciprocal similarity, then probably they are siblings, and are similar because they derive from the same area or areas. On the contrary, if their reciprocal similarity is higher than the one with the predecessors a *relatedEquivalent* relationship is more probable.

The metric $S(x,y)$ we propose as a measure of the similarity between two keywords in a corpus of document is designed to reward the non-trivial similarities that cannot be derived from the taxonomy:

$$S(x,y) = c(x,y) - w_{sa} c_{sa}(x,y) - w_{sub} |P(x|y) - P(y|x)|$$

where $c(x,y)$ is the cosine similarity between x and y, $c_{sa}(x,y)$ is the average cosine similarity with the common super-areas. $0<w_{sa}<1$ weighs the effects of the common super-areas on the similarity and in the evaluation will be set at 0.2; $0<w_{sub}<1$ weighs the importance of not having a subsumption relationship. In the evaluation $w_{sub}=0.2$.

The last part of the formula reduces the risk of inferring a *relatedEquivalent* relationship when there is actually a hierarchical one, by introducing a *malus* correlated with the difference of the subsumption probabilities $|P(x|y) - P(y|x)|$.

We infer a *relatedEquivalent* candidate when $S(x,y)>t_{re}$, where t_{re} is the threshold chosen by a human user. In the evaluation we used $t_{re}= 0.75$.

In those rare occasions in which for two keywords both a *relatedEquivalent* and a standard or temporal link can be inferred, it is up to the user to decide the priority. We decided to prefer the inference of the standard link rather than the *relatedEquivalent* one, and the *relatedEquivalent* rather than the temporal link.

The keywords which are found as suitable *relatedEquivalent* candidates are processed by a bottom-up single-linkage hierarchical clustering algorithm which uses the inverse of $S(x,y)$ as the distance between the elements.

The keywords in any resulting cluster are finally merged together in an aggregated keyword whose set of documents is the union of the sets of documents of all merged keywords. Since this new keyword should be inserted in the taxonomy, the process goes back to step one to start over again. The taxonomy will be considered complete and will be returned when no new *relatedEquivalent* relationships are inferred.

4 Evaluation

We used Klink to analyze a very large corpus of papers about the Semantic Web and related research areas. We needed a very big dataset that would offer challenges such as the presence of synonymous keywords to be merged after detecting a *relatedEquivalent* relationship among them and fuzzy keywords that might not be research areas. The collection of metadata available on MAS meets these requirements: moreover MAS offers useful APIs to provide access to their data.

As stated before, a rigorous analysis of a research area requires an unbiased sample of papers. Thus it would be inappropriate to take in consideration only the papers associated with the keyword "Semantic Web" or published in Semantic Web conferences. For this reason we constructed our corpus as follows: we first downloaded from MAS the metadata of 11,998 papers associated with the keyword "Semantic Web"; we then used this set to find the 120 research areas with which the "Semantic Web" has the highest number of co-occurrences and downloaded all the associated papers. The end results were 1,510,871 papers that we can consider to constitute an unbiased sample.

We tested different approaches to build a taxomomy, studying the impact of the different techniques presented in this paper on the final results. In particular we compared[14]:

1) The classic subsumption method [6] described in section 3.3 (labelled **S**);
2) The Klink approach to finding hierarchical standard links explained in section 3.3.1 (labelled **L**);
3) The Klink approach to finding hierarchical standard links with the integration of external knowledge described in section 3.3.3 (labelled **L+EXT**);
4) The full Klink algorithm, using both standard and temporal links (see section 3.3.2) with the integration of external knowledge (labelled **L+EXT+TL**).

The hypothesis was that Klink could be used to build taxonomies that are very similar, although not necessarily identical, to the ones created by a human user. Consistently with the discussion in section 2, the relationships inferred by our approach are instances of three kinds of semantic relationships: *broaderGeneric*, *contributesTo* and *relatedEquivalent*. However, as stated before, Klink is not able to distinguish between the first two relationships, characterizing both of them as hierarchical links. Hence, right now it is up to a human user to distinguish between these two types of hierarchical links, although we plan in future work to examine automatic ways to do so.

To evaluate the automatically built taxonomies we created a gold standard[15], which was passed on to three external experts for validation/revision. We started with a collection of the 120 keywords with the most numerous co-occurrences within the

[14] Because of space limitations, the only two parameters that we will analyze in this evaluation are the standard and temporal thresholds.

[15] The gold standard and the data generated by the algorithm are available at `http://kmi.open.ac.uk/technologies/rexplore/data/`

Semantic Web according to the MAS data. We then removed the less developed parts of the structure, e.g., the structure associated with the keyword "User Model", obtaining a final sample of 58 keywords: a reasonable size to be handled by the experts in their manual evaluation. About 15% of the relationships had to be changed to follow the directives of the experts. In about 7% of the cases the three experts disagreed on a relationship and we used the one suggested by two out of three. We chose to use a gold standard since it allowed to test not only the final version of the algorithm but also to study the different contributions offered by its parts as a function of the thresholds.

We selected two taxonomies with different degrees of focus on the Semantic Web. The first one (labeled Set1) covers “Semantic Web” together with “Formal Ontology” and “Knowledge Representation”. The second one (labeled Set 2) includes also the other areas and is expected to yield inferior results since the sample does not cover entirely those areas.

We ran the algorithms and compared the generated taxonomy with the gold standard by computing the recall and the precision of the inferred relationships and their harmonic mean (F-measure). To reduce complexity, we set the standard and the temporal link threshold at the same value.

Figure 1 shows the relation between precision and recall obtained with the four algorithms **S**, **L**, (**L+EXT**) and (**L+EXT+LT**) for the two sets.

Using the proposed metric for inferring hierarchical relationship described in section 3.3.1, **L** yields a recall of 53% for Set1 and 38% for Set2 for a precision higher than 80%. The basic subsumption method **S** found in the literature yields for the same level of precision a recall of 30% in Set1 and 8% in the Set2.

The integration of the external knowledge (**L+EXT**) appears to be effective allowing, with t=0.15, a precision of 93% and a recall of 90% for Set1 and precision 64% with recall 84% for Set2. With t=0.2, precision and recall go respectively to 95% and 69% in Set1 (78% and 64% for Set2).

The temporal links are able to improve the results even more, especially for Set2, where more difficulties are posed by the chore of inferring subtrees in areas for which we do not have the complete structure. With threshold 0.2, (**L+EXT+LT**) boosts the recall to 92% with a precision 94% for Set1 (86% and 78% for Set2). By raising the threshold to 0.25, precision reaches 98% with recall 73% (88% and 73% for Set2). Figure 2 shows the F-measure for the two sets as a function of the threshold.

All results agree in indicating **L+EXT+TL** as the best approach, followed by **L+EXT** and farther away by **L**. To statistically evaluate the differences among the curves in Figures 1 and 2, we employed the chi-square test. The comparison of precision between **L+EXT+TL** and **L** yields p=2x10^{-3} for Set1 and p=2x10^{-5} for Set2; both statistically significant. The comparison between **L+EXT+TL** and **L+EXT** yields no statistically significant differences for Set1, whereas p=9x10^{-3} for Set2. Similar results are obtained for recall. The fact that a statistically significant difference between **L+EXT+ST** and **L+EXT** exists only for Set2 indicates that the insertion of the temporal link was determinant for inferring relationships in a context where the set of keywords related to some research area is not complete.

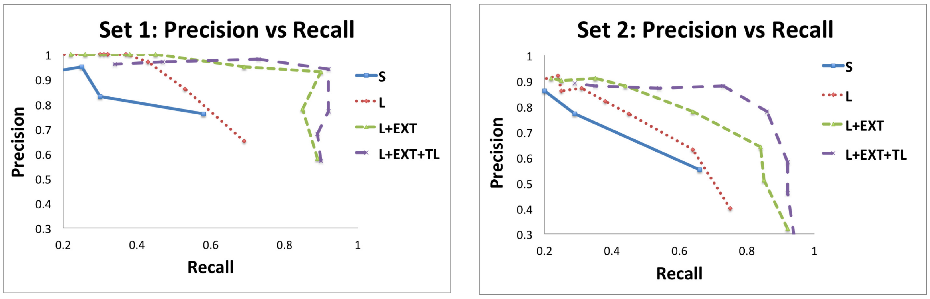


Fig. 1. Precision vs Recall for Set 1 and Set 2

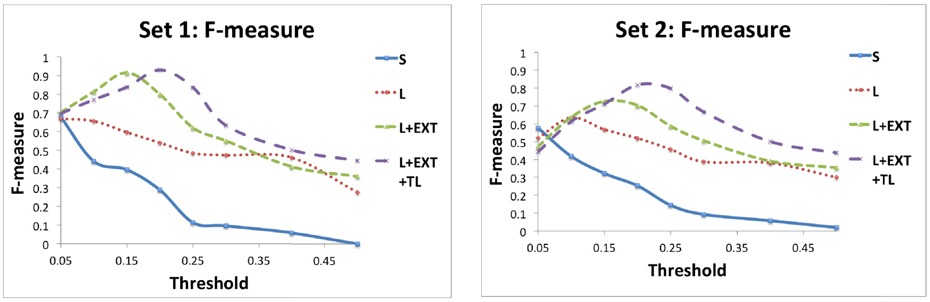


Fig. 2. F-measure for Set 1 and Set 2

The indication on the threshold places t between 0.15 and 0.25, depending on the desired trade off between precision and recall.

Figure 3 and Figure 4 compare the fraction of the gold standard representing the Semantic Web with the automatically generated version using the **L+EXT+LT** version of the algorithm and t=t_t=0.2.

The few discrepancies between the two figures open interesting perspectives. As an example, let us consider the relationship between the areas of Web of Data and of Linked Data. Before the test runs, two of our experts considered correct to have Web of Data as *skos:broaderGeneric* than Linked Data, whereas the third preferred *relatedEquivalent*. Our algorithm appears to have chosen a third possibility, i.e., that Web of Data is more specific than Linked Data. In our dataset the papers associated with Web of Data were also associated with Linked Data in 54% of the cases, while the contrary happens only in 7.5% of the cases. The very high value of the cosine similarity, 0.94, between the two keywords indicated a strong relationship and the co-occurrence analysis suggested that the super-area should be Linked Data. Hence, the data appear to indicate that in practice authors use "Linked Data" as a generic term for talking about the web of data to a much greater extent than they use the term "Web of Data".

While experts may consider this perspective incorrect, a data-driven, bottom-up approach like ours can also be used to highlight these interesting discrepancies between the intended coverage of research areas and the mental models that emerge from the way these terms are used in.

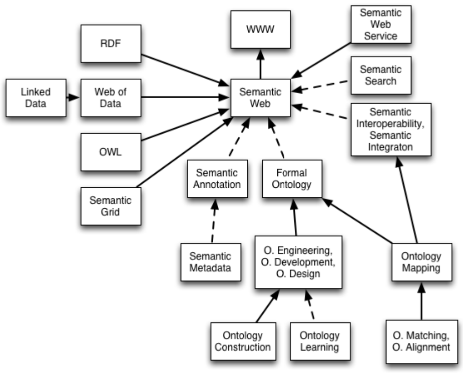


Fig. 3. A portion of the gold standard associated with the Semantic Web. The solid arrows represent *broaderGeneric* relationships, the dashed ones *contributesTo* ones.

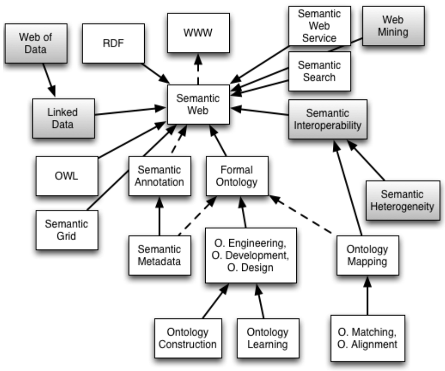


Fig. 4. A portion of the automatically generated taxonomy associated with the Semantic Web. The solid arrows represent standard links, the dashed ones temporal ones.

5 Related Work

Taxonomies can be very useful tools for improving search results and their presentation [8]. In particular these structures are helpful in faceted search [9], which is the search paradigm based on the indexing of documents along multiple orthogonal taxonomies. Faceted semantic search systems tend instead to use ontological relationships, such as *partOf* or *isA* [10].

To build manually a taxonomy is however not a trivial task and the human crafted ones may not mirror the true internal relationships of a corpus of documents, but rather the standard taxonomy of the field. For this reason to automatically generate a taxonomy from a corpus of document or metadata is an important challenge.

Traditionally there have been two different approaches to this task. The first is based on clustering techniques [11], the second, developed especially in computational linguistic, rests on the detection of lexico-syntactic patters [12].

The *TaxGen* framework [13] uses a hierarchical agglomerative clustering algorithm and text mining techniques for the creation of a taxonomy from a collection of unstructured documents. In [14] a hierarchical clusterization algorithm is applied on web pages and a top-down partitioning is used to generate a multi-way-tree taxonomy from the binary tree. Our method also exploits hierarchical algorithms and similarity distances between keywords. However they are not used for generating the hierarchical structure but for merging the keywords for which a *relatedEquivalent* relationship was inferred.

The other traditionally used method is based on the detection of linguistic patterns that appear in a corpus of documents. In [15] an approach called Lexico-Syntactic Pattern Extraction (LSPE) is presented, which exploits patterns like "such as…" and

"and other…" to discover relationships between terms. A similar procedure is also reported in [16] where a clustering-based sense disambiguation heuristics is proposed for pruning the resulting taxonomy. The same technique can be used also to infer ontological relationships like *subClassOf*, as in [17].

The work of Sanderson and Croft [6] proposes an approach that allows generating automatically concept hierarchies without the use of training data or clustering techniques. Namely they use the probability that a keyword is associated with another to infer subsumptions, as discussed in section 3.3. The same idea is extended in the GrowBag algorithm [18], which exploits the second order co-occurrence made explicit by a biased *PageRank* algorithm.

The basic idea that we have used to infer the subsumption relationship between keywords is similar to the one found in [6]. However we extended this approach i) by introducing a set of very different metrics, which exploit cosine similarities and temporal dimensions and ii) by integrating external knowledge into the process.

Other authors have proposed the use of external knowledge from web pages for finding hierarchical relationships. In [7] three heuristic techniques are suggested for mining topic-specific knowledge, however such methods need specific patterns that may not be very common in all domains.

The approach proposed in this paper can find practical applications in the growing areas of academic repositories (see for example [3], [19]), to support users in the exploration and use of such repositories. In particular, it can be seen as a complementary approach to techniques employed for managing, cleaning and organizing folksonomies of tags attached to research papers, as notably applied in the Bibsonomy system [20]. The semantic GrowBag algorithm already mentioned was similarly employed to derive automatically facets to be used in the faceted browsing of large publication collections (in Faceted DBLP, see [18]). Our approach however focuses on finding relationships between keywords with a high level of accuracy, which are verified as corresponding to research areas. As a result we can provide a robust navigational structure for collections of research publications, while reducing the need for manually curating the structure of the collection.

Other related works concern complementary approaches, which investigate the connections between authors of papers (the network of researchers), in order to establish relationships in their areas of interest (see for example [21]). The results obtained naturally differ in their views of the types of relationships shared between research areas/communities.

6 Conclusions

We have presented Klink, a novel algorithm to infer relationships between keywords from a collection of terms associated with documents. Klink was tested on a large corpus of data from MAS to analyze the relationships between the Semantic Web research area and other related areas.

The results of the evaluation shows a statistically significant improvement of the performance by using Klink over the classic subsumption method: the values of recall and precision obtained in regard to the gold standard are highly satisfactory. The keyword-centered perspective of the algorithm also offers interesting opportunities for analyzing situations in which the experts do not agree on the kinds of relationships between two research areas.

The next steps include three main avenues of work. Firstly, we are currently developing a novel system, called *Rexplore*, whose aim is to improve the support available to users to explore and make sense of research data, by integrating a wide range of novel visualization methods. The method presented in this paper will be integrated with Rexplore, to support a more powerful and flexible way to map research areas to authors and publications. The second avenue of work focuses on developing new methods to automatically distinguish between *broaderGeneric* and *contributesTo* relationships, to avoid the need for humans to perform this final semantic step. Finally we want to improve our algorithm by allowing it to recognize sets of keywords of similar meaning that fall under a common area, even when this is not explicitly present in the keyword collection. For example, both RDF and OWL can be seen as sub-areas of a more generic area, which could be called "Web Knowledge Representation", however such area is rarely used as a keyword by authors. Again, by using a combination of machine learning techniques and external knowledge, we are confident that a method can be developed, which will be able to handle these situations correctly.

References

1. 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.) ISWC/ASWC 2007. LNCS, vol. 4825, pp. 802–815. Springer, Heidelberg (2007)
2. Latif, A., Afzal, M.T., Helic, D., Tochtermann, K., Maurer, H.: Discovery and Construction of Authors' Profile from Linked Data (A case study for Open Digital Journal). In: WWW 2010 Workshop on Linked Data on the Web (LDOW 2010), vol. 628. CEUR-WS, Raleigh, North Carolina, USA (2010)
3. Glaser, H., Millard, I.: Knowledge-Enabled Research Support: RKBExplorer.com. In: Proceedings of Web Science 2009, Athens, Greece (2009)
4. Stankovic, M., Rowe, M.: Mapping Tweets to Conference Talks: A Goldmine for Semantics. In: ISWC 2010 Workshop on Social Data on the Web, Shanghai, China (2010)
5. Benjamins, R., Fensel, D., Decker, S.: KA2: Building Ontologies for the Internet: A Midterm Report. International Journal of Human-Computer Studies 51(3) (1999)
6. Sanderson, M., Croft, B.: Deriving concept hierarchies from text. In: Proceedings of the SIGIR Conference, pp. 206–213 (1999)
7. Liu, B., Chin, C.W., Ng, H.T.: Mining topic-specific concepts and definitions on the web. In: Proceedings of WWW 2003, pp. 251–260. ACM, New York (2003)
8. Pratt, W., Hearst, M.A., Fagan, L.M.: A knowledge-based approach to organizing retrieved documents. In: AAAI Conference, Menlo Park, CA, USA (1999)

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. Suominen, O., Viljanen, K., Hyvänen, E.: User-Centric Faceted Search for Semantic Portals. In: Franconi, E., Kifer, M., May, W. (eds.) ESWC 2007. LNCS, vol. 4519, pp. 356–370. Springer, Heidelberg (2007)
11. Assadi, H.: Construction of a regional ontology from text and its use within a documentary system. In: Guarino, N. (ed.) Formal Ontology in Information Systems, Proceedings of FOIS 1998, Trento, Italy, pp. 236–249 (1999)
12. Morin, E.: Automatic acquisition of semantic relations between terms from technical corpora. In: Proceedings of the 5th International Congress on Terminology and Knowledge Engineering (1999)
13. Müller, A., Dorre, J.: The TaxGen Framework: Automating the Generation of a Taxonomy for a Large Document Collection. In: Proceedings of the 32nd Hawaii International Conference on System Sciences, vol. 2, pp. 20–34 (1999)
14. Chuang, S., Chien, L.: A practical web-based approach to generating topic hierarchy for text segments. In: Proceedings of the 13th ACM Conference on Information and Knowledge Management, Washington, D.C., USA (2004)
15. Hearst, M.: Automated discovery of WordNet relations. In: Fellbaum, C. (ed.) WordNet: An Electronic Lexical Database, pp. 131–153. MIT Press (1998)
16. Recio-Garcia, J.A., Wiratunga, N.: Taxonomic Semantic Indexing for Textual Case-Based Reasoning. In: Bichindaritz, I., Montani, S. (eds.) ICCBR 2010. LNCS, vol. 6176, pp. 302–316. Springer, Heidelberg (2010)
17. De Cea, G., de Mon, I., Montiel-Ponsoda, E.: From Linguistic Patterns to Ontology Structures. In: 8th Conference on Terminology and Artificial Intelligence (2009)
18. Diederich, J., Balke, W., Thaden, U.: Demonstrating the Semantic GrowBag: Automatically Creating Topic Facets for FacetedDBLP. In: Proceedings of JCDL 2007. ACM, New York (2007)
19. Jaschke, R., Grahl, M., Hotho, A., Krause, B., Schmitz, C., Stumme, G.: Organizing Publications and Bookmarks in BibSonomy. In: WWW Workshop on Social and Collaborative Construction of Structured Knowledge (2007)
20. Benz, D., Hotho, A., Jäschke, R., Krause, B., Mitzlaff, F., Schmitz, C., Stumme, G.: The social bookmark and publication management system bibsonomy. VLDB Journal 19(6), 849–875 (2010)
21. Krafft, D., Cappadona, N., Caruso, B., Corson-Rikert, J., Devare, M., Lowe, B.: VIVO: Enabling National Networking of Scientists. In: Proceedings of the Web Science Conference 2010, Raleigh, US, pp. 1310–1313 (2010)

Discovering Concept Coverings in Ontologies of Linked Data Sources

Rahul Parundekar, Craig A. Knoblock, and José Luis Ambite

Information Sciences Institute and Department of Computer Science
University of Southern California
4676 Admiralty Way, Suite 1001, Marina del Rey, CA 90292
{parundek,knoblock,ambite}@usc.edu

Abstract. Despite the increase in the number of linked instances in the Linked Data Cloud in recent times, the absence of links at the concept level has resulted in heterogenous schemas, challenging the interoperability goal of the Semantic Web. In this paper, we address this problem by finding alignments between concepts from multiple Linked Data sources. Instead of only considering the existing concepts present in each ontology, we hypothesize new composite concepts defined as disjunctions of conjunctions of (RDF) types and value restrictions, which we call *restriction classes*, and generate alignments between these composite concepts. This extended concept language enables us to find more complete definitions and to even align sources that have rudimentary ontologies, such as those that are simple renderings of relational databases. Our concept alignment approach is based on analyzing the extensions of these concepts and their linked instances. Having explored the alignment of conjunctive concepts in our previous work, in this paper, we focus on concept coverings (disjunctions of *restriction classes*). We present an evaluation of this new algorithm to Geospatial, Biological Classification, and Genetics domains. The resulting alignments are useful for refining existing ontologies and determining the alignments between concepts in the ontologies, thus increasing the interoperability in the Linked Open Data Cloud.

1 Introduction

The Web of Linked Data has grown significantly in the past few years – 31.6 billion triples as of September 2011. This includes a wide range of data sources from the government (42%), geographic (19.4%), life sciences (9.6%) and other domains.[1] A common way that the instances in these sources are linked to others is through the *owl:sameAs* property. Though the size of Linked Data Cloud is increasing steadily (10% over the 28.5 billion triples in 2010), inspection of the sources at the ontology level reveals that only a few of them (15 out of the 190 sources) include mappings between their ontologies. Since interoperability is crucial to the success of the Semantic Web, it is essential that these heterogenous schemas, the result of a de-centralized approach to the generation of data

[1] http://www4.wiwiss.fu-berlin.de/lodcloud/state/

and ontologies, also be linked. The problem of schema linking, such as schema matching in databases and ontology alignment in the Semantic Web, has been well researched [5,1,4]. As in Instance-based Matching [4,3,7], we follow an extensional approach to generating the alignments. The novelty of our approach consists of generating new concept hypotheses beyond the concepts originally present in the ontologies, and aligning these extended concepts by exploiting the linked instances in the Linked Data Cloud.

The problem of finding alignments in ontologies of Linked Data sources is non-trivial, since there might not be one-to-one concept equivalences. In some sources the ontology is extremely rudimentary, for example *GeoNames* has only one class - *geonames:Feature*, and the alignment of such an ontology with a well defined one, such as *DBpedia*, is not particularly useful. In order to be successful in linking ontologies, we need to generate more expressive concepts. The necessary information to do this is often present in the properties and values of the instances in the sources. For example, in *GeoNames* the values of the *featureCode* and *featureClass* properties provide useful concept constructors, which can be aligned with existing concepts in *DBpedia*, so that we have that the concept *geonames:featureCode=P.PPL* (populated place) aligns to *dbpedia:City*. Therefore, our approach explores the space of concepts defined by value restrictions, which we will call *restriction classes* in the reminder of the paper. A *value restriction* is a concept constructor present in expressive description logics, such as OWL-DL ($\mathcal{SHOIN}(\mathcal{D})$) [6]. We consider class assertions (rdf:type) and value restrictions on both object and data properties, which we will represent uniformly as $\{p = v\}$, where either p is an object property and v is a resource (including rdf:type=Class), or p is a data property and v is a literal. We consider two *restriction classes* equal if their respective instance sets can be identified as equal after following the *owl:sameAs* links.

In our previous work [10], we explored conjunctive *restriction classes*. In this paper, we explore disjunctive *restriction classes*. Specifically, we focus on concept coverings where a larger concept from one source can be explained by (i.e., is extensionally equivalent to) the union of multiple smaller classes in the other source. Our approach finds alignments based on the extensions of the concepts, that is, the sets of instances satisfying the definitions of the restriction classes. We believe that this is an important feature of our approach in that it allows one to understand the relationships in the *actual* linked data and their corresponding ontologies. The alignments generated can readily be used for modeling and understanding the sources since we are modeling what the sources actually contain as opposed as to what an ontology disassociated from the data appears to contain based on the class name or description.

This paper is organized as follows. First, we describe the Linked Open Data sources that we align. Second, we present the alignment algorithm that consists of two steps: finding initial equivalence and subset relations, and then discovering *concept coverings* using disjunctions of *restriction classes*. Third, we describe representative alignments discovered by our approach and present an evaluation of the results. An interesting outcome of our algorithm is that it identifies inconsistencies and possible errors in the linked data, and provides a method for automatically curating the Linked Data Cloud. Finally, we compare against related work, and discuss our contributions and future work.[2]

2 Sources Used for Alignments

In the Linked Open Data Cloud, sources often conform to different, but related, ontologies that can also be meaningfully linked [2,9,10]. In this section we describe some of these sources from different domains that we align, instances in which are linked using an equivalence property like *owl:sameAs*.

Linking *GeoNames* with Places in *DBpedia*: *DBpedia* (dbpedia.org) is a knowledge base that covers multiple domains including around 526,000 places and other geographical features from the Geospatial domain. We align concepts in *DBpedia* with *GeoNames* (geonames.org), which is a geographic source with about 7.8 million geographical features. GeoNames uses a rudimentary flat-file like ontology, where all instances belong to a single concept of *Feature*, with the type data (e.g. mountains, lakes, etc.) encoded in the *featureClass and featureCode* properties.

Linking *LinkedGeoData* with Places in *DBpedia*: We also find alignments between the ontologies behind *LinkedGeoData* (linkedgeodata.org) and *DBpedia*. *LinkedGeoData* is derived from the *Open Street Map* initiative with around 101,000 instances linked to *DBpedia* using the *owl:sameAs* property.

Linking Species from *Geospecies* with *DBpedia*: The *Geospecies* knowledge base (lod.geospecies.org) contains a taxonomic classification of living organisms linked to species in *DBpedia* using the *skos:closeMatch* property. Since these sources have many species in common, they are ideal for finding alignments between the vocabularies.

Linking Genes from *GeneID* with *MGI*: The Bio2RDF (bio2rdf.org) project contains inter-linked life sciences data extracted from multiple data-sets that cover genes, chemicals, enzymes, etc. We consider two sources from the Genetics domain from Bio2RDF, *GeneID* (extracted from the National Center for Biotechnology Information database) and *MGI* (extracted from the Mouse Genome Informatics project), where the genes are marked equivalent.

In Section 4 we provide results for the four alignment experiments described above. In the rest of this paper we explain our methodology, which is source independent, by using the alignment of *GeoNames* with *DBpedia* as an example.

3 Finding Concept Coverings across Ontologies

We use a two step approach to find *concept coverings*. First, we extract atomic equivalent and subset alignments from the two sources where the *restriction*

[2] This paper is an extended version of our workshop paper [11]. We have added more formal descriptions, explanations of the algorithms and detailed evaluation.

class on each side contains a single *property-value pair* and no conjunction or disjunctions. These are the simplest alignments that can be defined. We then use these to find concept coverings by describing a larger concept with a union of smaller ones using set containment.

We also discuss how our alignment approach detects outliers, which often indicate missing or incorrect links, and provides a powerful tool to curate the Linked Data cloud.

3.1 Finding Alignments with Atomic *Restriction Classes*

As a precursor to finding *concept coverings* between the two sources, our algorithm first finds alignments where the *restriction classes* on each side of the alignment are atomic - i.e. have one *property-value pair* each. In our previous work [10], we used a similar approach to find alignments between the ontologies where a conjunction of *restriction classes* was aligned with its equivalent concept in the other source. In this paper we focus on atomic *restriction classes*. Since we do not need to find alignments of conjunctive *restriction classes*, the search space is polynomial rather than combinatorial.

The sources are first prepared for exploration by performing an inner join on the equivalence property (e.g., *owl:sameAs*) and optimized by removing inverse-functional properties. Then, the following algorithm is used to find the alignments between atomic *restriction classes*.

```
for all $p_1$ in $Source_1$ and distinct $v_1$ associated with $p_1$ do
    $r_1 \leftarrow$ restriction class $\{p_1 = v_1\}$ containing all instances where $p_1 = v_1$
    $Img(r_1) \leftarrow$ Find all corresponding instances from $Source_2$ to those in $r_1$,
linked by owl:sameAs
    for all $p_2$ in $Source_2$ and distinct $v_2$ associated with $p_2$ do
        $r_2 \leftarrow$ restriction class $\{p_2 = v_2\}$ containing all instances where $p_2 = v_2$
        $P \leftarrow \frac{|Img(r_1) \cap r_2|}{|r_2|}$, $R \leftarrow \frac{|Img(r_1) \cap r_2|}{|r_1|}$
        if $P \geq \theta$ then $alignment(r_1, r_2) \leftarrow r_1 \subset r_2$
        end if
        if $R \geq \theta$ then $alignment(r_1, r_2) \leftarrow r_2 \subset r_1$
        end if
        if $P \geq \theta$ and $R \geq \theta$ then $alignment(r_1, r_2) \leftarrow r_1 \equiv r_2$
        end if
    end for
end for
```

Fig. 1 illustrates the set comparison operations of our algorithm. In order to allow a certain margin of error induced by the data set, we use $P \geq \theta$ and $R \geq \theta$ (instead of $P = 1$ and $R = 1$, which would hold if there were no error or missing links) in our score function. In our experiments we used a threshold $\theta = 0.9$, which was determined empirically, but can be changed as desired. For example, consider the alignment between *restriction classes* $\{$*geonames:countryCode*=ES$\}$

from *GeoNames* and {*dbpedia:country* = *dbpedia:Spain*} from *DBpedia*. Based on the extension sets, our algorithm finds $|Img(r_1)| = 3198$, $|r_2| = 4143$, $|Img(r_1) \cap r_2| = 3917$, $R' = 0.9997$ and $P' = 0.9454$. Thus, the algorithm considers the alignment as equivalent in an extensional sense. Some alignments that do not qualify as equivalent, but with the smaller concept contained in the larger concept, qualify as subset relations. For example, we find that each of {*geonames:featureCode* = *S.SCH*}, {*geonames:featureCode* = *S.SCHC*} and {*geonames:featureCode* = *S.UNIV*} (i.e. Schools, Colleges and Universities from *GeoNames*) are subsets of {*dbpedia:EducationalInstitution*}.

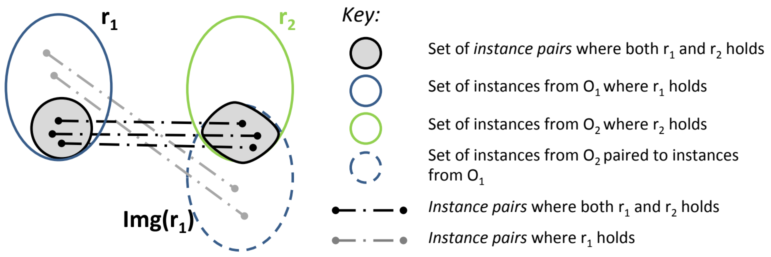

r₁
r₂
Img(r₁)
Key:
Set of instance pairs where both r₁ and r₂ holds
Set of instances from O₁ where r₁ holds
Set of instances from O₂ where r₂ holds
Set of instances from O₂ paired to instances from O₁
Instance pairs where both r₁ and r₂ holds
Instance pairs where r₁ holds

Fig. 1. Comparing the linked instances from two ontologies

Similar to our previous work [10], we also use certain optimization strategies for faster computation. For example, if we explore the properties lexicographically, the search space is reduced to half because of symmetry. To qualify as a concept, the intersection of the *restriction classes* needs to have a minimum support, which we set experimentally to ten instances.

3.2 Identifying *Concept Coverings*

In step two, we use the subclasses and equivalent alignments generated by the previous step to try and align a larger concept from one ontology with a union of smaller subsumed concepts in the other ontology. To define a larger concept, we group its subclasses from the other source that have a common property and check whether they are able to cover the larger concept. By keeping the larger *restriction class* atomic and by grouping the smaller *restriction classes* with a common property, we are able to find intuitive definitions while keeping the problem tractable. The disjunction operator that groups the smaller *restriction classes* is defined such that *i)* the concept formed by the disjunction of the classes represents the union of their set of instances, *ii)* the property for all the *property-value pairs* of the smaller aggregated classes is the same. We then try to detect the alignment between the larger concept and the union *restriction class* by using an extensional approach similar to the previous step. The algorithm for generating the hypotheses and the alignments is as follows:

```
for all alignments found in the previous step, with larger concepts from one source with multiple subclasses from the other source do
    $U_L \leftarrow$ larger restriction class$\{p_L = v_L\}$, and corresponding instances.
    for all smaller concepts grouped by a common property ($p_S$) do
        $U_S \leftarrow$ the union restriction class, and the corresponding instances of all the smaller restriction classes$\{p_S = \{v_1, v_2, ...\}\}$
        $U_A \leftarrow Img(U_L) \cap U_S$, $P_U \leftarrow \frac{|U_A|}{|U_S|}$, $R_U \leftarrow \frac{|U_A|}{|U_L|}$
        if $R_U \geq \theta$ then $alignment(r_1, r_2) \leftarrow U_L \equiv U_S$
        end if
    end for
end for
```

Since all smaller classes are subsets of the larger *restriction class*, $P_U \geq \theta$ by construction. We used $\theta = 0.9$ in our experiments. The smaller *restriction classes* that were omitted in the first step because of insufficient support size of their intersections, were included in constructing U_S for completeness.

Figure 2 provides an example of the approach. The first step is able to detect that alignments such as {*geonames:featureCode* = *S.SCH*}, {*geonames:featureCode* = *S.SCHC*}, {*geonames:featureCode* = *S.UNIV*} are subsets of {*rdf:type* = *dbpedia:EducationalInstitution*}. As can be seen in the Venn diagram in Figure 2, U_L is *Img*({*rdf:type* = *dbpedia:EducationalInstitution*}), U_S is {*geonames:featureCode* = *S.SCH*} ∪ {*geonames:featureCode* = *S.SCHC*} ∪ {*geonames:featureCode* = *S.UNIV*}, and U_A is the intersection of the two. Upon calculation we find that R'_U for the alignment of *dbpedia:EducationalInstitution* to {*geonames:featureCode*= {S.SCH, S.SCHC, S.UNIV}} is 0.98. We can thus confirm the hypothesis and consider U_L & U_S equivalent. Section 4 describes these calculations and additional examples of *concept coverings*.

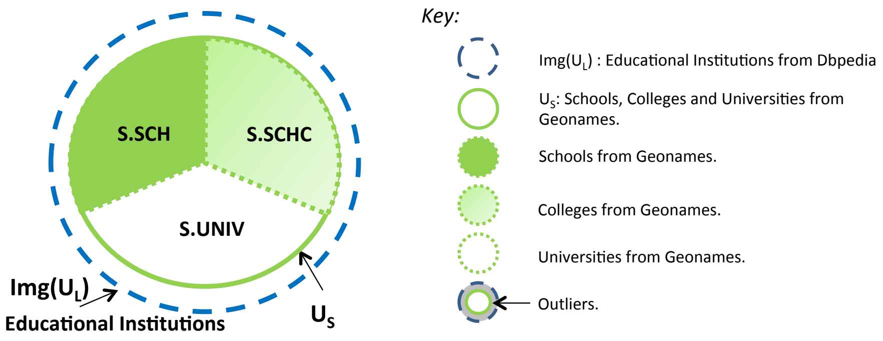

S.SCH
S.SCHC
S.UNIV
Img(U_L)
Educational Institutions
U_S
Key:
Img(U_L) : Educational Institutions from Dbpedia
U_S: Schools, Colleges and Universities from Geonames.
Schools from Geonames.
Colleges from Geonames.
Universities from Geonames.
Outliers.

Fig. 2. Concept covering of Educational Institutions from *DBpedia*

3.3 Curating the Linked Data Cloud

It turns out that the outliers, the instances of the *restriction classes* that do not satisfy subset relations despite the error margins, are often due to incorrect and missing links or assertions. We are able to detect these, thus providing a novel method to curate the Web of Linked Data.

In the alignment of {*rdf:type = dbpedia:EducationalInstitution*} to {*geonames: featureCode* = {S.SCH, S.SCHC, S.UNIV}} we find 8 outliers (Table 6, row 1). For {*rdf:type = dbpedia:EducationalInstitution*}, 396 instances out of the 404 Educational Institutions were accounted for as having their *geonames:featureCode* as one of *S.SCH, S.SCHC or S.UNIV*. From the 8 outliers, 1 does not have a *geonames:featureCode* property asserted. The other 7 have their feature codes as either S.BLDG (3 buildings), S.EST (1 establishment), S.HSP (1 hospital), S.LIBR (1 library) or S.MUS (1 museum). This case requires more sophisticated curation and the outliers may indicate a case for multiple inheritance. For example, the hospital instance in geonames may be a medical college that could be classified as a university.

In the {*dbpedia:country = Spain*} ≡ {*geonames:countryCode* = {ES}} alignment (Table 6, row 2), one outlier instance was identified as having the country code IT (Italy) in *GeoNames*, suggesting an incorrect link/assertion. The algorithm was able to flag this situation as a possible error, since there is overwhelming support for 'ES' being the country code of Spain. Our union alignment algorithm is able to detect similar other outliers and provides a powerful tool to quickly focus on links that require human curation, or that could be automatically flagged as problematic, and provides evidence for the error.

4 Experimental Results

The results of our *concept covering* algorithm over the four pairs of sources we consider appear in Table 1. The first step of our algorithm was able to generate about 180k equivalence and subset alignments. After running the covering algorithm, 77966 subset alignments were explained by 7069 coverings, for a compression ratio of about 11:1.

Table 1. Concept Coverings Found in the 4 Source Pairs

$Source_1$	$Source_2$	O_1-O_2: Coverings (Subset Alignments)	O_2-O_1 Coverings (Subset Alignments)	Total Coverings
GeoNames	*DBpedia*	434 (2197)	318 (7942)	752
LinkedGeoData	*DBpedia*	2746 (12572)	3097 (48345)	5843
Geospecies	*DBpedia*	191 (1226)	255 (2569)	446
GeneID	*MGI*	6 (29)	22 (3086)	28

4.1 Representative Examples of the Concept Coverings Found

Some representative examples of the *concept coverings* found are shown in Tables 6, 7 and 8. In the tables, for each *concept covering*, column 2 describes the large *restriction class* from *ontology*$_1$ and column 3 describes the union of the (smaller) classes on *ontology*$_2$ with the corresponding property and value set. The score of the union is noted in column 4 ($R_U = \frac{|U_A|}{|U_L|}$) followed by $|U_A|$ and $|U_L|$ in columns 5 and 6. Column 7 describes the outliers, i.e. values v_2 of property p_2 that form *restriction classes* that are not direct subsets of the larger *restriction class.* Each of these outliers also has a fraction with the number of instances that belong to the intersection over the the number of instances of the smaller *restriction class* (or $\frac{|Img(r_1) \cap r_2|}{|r_2|}$). One can see that the fraction is less than our relaxed subset score. If the value of this fraction was greater than the relaxed subset score (i.e. $\theta = 0.9$), the set would have been included in column 3 instead. The last column mentions how many of the total U_L instances were we able to explain using U_A and the outliers. For example, the *concept covering* #1 of Table 6 is the Educational Institution example described before. It shows how educational institutions from *DBpedia* can be explained by schools, colleges and universities in *GeoNames.* Column 4, 5 and 6 explain the alignment score R_U (0.98), the size U_A (396) and the size of U_L (404). Outliers (S.BLDG, S.EST, S.LIBR, S.MUS, S.HSP) along with their P' fractions appear in column 7. Thus, 403 of the total 404 instances were identified as either part of the covering or the outliers (see column 8). The remaining instance did not have a *geonames:featureCode* property asserted.

In some of the *concept coverings* discovered, the alignments found were intuitive because of an underlying hierarchical nature of the concepts involved, especially in case of alignments of administrative divisions in geospatial sources and alignments in the biological classification taxonomy. For example, #3 highlights alignments that reflect the containment properties of administrative divisions. Other interesting types of alignment were also found. For example #7 tries to map two non-similar concepts. It explains the license plate codes found in the state (bundesland) of Saarland. For space, we explain the other *concept coverings* inside Tables 6, 7 and 8. The complete set of alignments discovered by our algorithm is available online.[3]

Outliers. In alignments, we also found inconsistencies, identified by three main reasons: **(i)***Incorrect instance alignments* - outliers arising out of possible erroneous equivalence link between instances (e.g., in #4, a hill is linked to an airport, etc.), **(ii)** *Incorrect values for properties* - outliers arising out of possible erroneous assertion for a property (e.g. #5, #6, Flags of countries appear as values for the *country* property). In the tables, we also mention the classes that these inconsistencies belong to along with their support. We are unable to detect correct alignments if there is insufficient support for coverage due to missing links between instances or missing instances (e.g. in #9 we find a complete coverage with all instances, but it is incomplete – the state of New Jersey has 21 counties).

[3] `http://www.isi.edu/integration/data/UnionAlignments`

4.2 Evaluation

We present an evaluation of a random set of 642 discovered alignments across the tested source pairs to describe the precision of our approach. We checked the correctness of each of the 642 alignments manually, after verifying the completeness of *concept coverings* on websites with the relevant information. Precision was calculated as the ratio of alignments marked correct to the size of the random set. Establishing recall is difficult as finding the ground truth of all possible *concept coverings* is infeasible due to the large size of the sources and the combinatorial nature of the disjuntions. We do, however, provide an evaluation of the country alignments found in terms of precision and recall as an example.

Linking *GeoNames* with places in *DBpedia*: As shown in Table 2, out of the 752 (i.e. 434 + 328) alignments found between *GeoNames* and *DBpedia*, we evaluated 236 (i.e. 185 + 51) alignments. 152 (i.e. 127 + 25) of them were found to be correct after resolving redirects (synonyms in *DBpedia*), giving a precision of 64.40%, while 84 alignments were found to be incorrect. These 84 alignments were found to suffer common patterns of error. There are 40 alignments that had incorrect assertions of their properties. For example, in many instances in *DBpedia*, the `county` property assertion was misspelled as `country` (especially for places in UK & Ukraine), or the ".svg" file of the flag of a country appeared *dbpedia:country* value. The corresponding alignments, which we counted as incorrect, could have been properly detected if the data was cleaner. We detected only partial alignments for 14 others, where the smaller concepts left out were incorrectly classified as outliers due to insufficient support ($R < 0.9$). There were 7 partial alignments that were incorrectly detected as complete ($R > 0.9$), similar to the New Jersey example mentioned earlier. Another 14 alignments suffered from a mismatch to a parent, because of insufficient links/instances. The remaining 9 alignments had an assortment of problems in the values of properties. For example, regions inside a country (Andean Region of Colombia) appeared as value for the country property (Colombia).

Precision, recall and f-measure of Country Alignments: Since manually establishing ground truth for all possible concept coverings in the four sources is infeasible, we decided to find the precision and recall of only the country alignments we found, as an illustration. These are alignments having a common pattern, aligning a *restriction class* with a *dbpedia:country* property with other *restriction classes* featuring *geonames:countryCode* property or vice-versa. A ground truth was established by manually checking what possible country alignments were present in the two sources. Even then, establishing the ground truth needed some insight. For example, Scotland, England, Wales, Northern Ireland & the United Kingdom are all marked as countries in *DBpedia*, while in *GeoNames*, the only corresponding country is the United Kingdom. In cases like these, we decided to relax the evaluation constraint of having and alignment with a country from either of these, as correct. Another similar difficulty was in cases where militarily occupied territories were marked as countries (e.g. Golan Heights occupied by Israel is marked as *dbpedia:country*).

Table 2. Linking *GeoNames* with places in *DBpedia*

Description of Pattern Observed	Alignments w/ larger class from *GeoNames*	Alignments w/ larger class from *DBpedia*
Total # Alignments	434	328
# Alignments Evaluated	**185**	**51**
Correct (after resolving redirects)	**127**	**25**
Unidentified due to mislabelling the Country property as County	5	
Unidentified due to '.svg' file of the flag as value for the country	35	
Partially found with remaining as outliers	3	11
Partially found without outliers		7
Misaligned with a parent concept	8	6
Other problems	7	2

Out of the 63 country alignments detected, 26 were correct. 27 other alignments had a '.svg' file appearing as value of the country property in *DBpedia*. We would have detected such *concept coverings*, had such assertions for the country property been correct. Since this is a problem with the data and not our algorithm, we consider these 27 as correct for this particular evaluation. We thus get a precision of 84.13% ((26+27) out of 63). The two sources contained around 169 possible country alignments between them, including countries with a '.svg' value for the country property. There were many alignments in the ground truth that were not found because the system did not have enough support ($R < 0.9$) to pass our threshold. Accordingly, the recall was 31.36% and the F-measure was 45.69%.

Linking *LinkedGeoData* with places in *DBpedia*: We evaluated 200 alignments found between *LinkedGeoData* and *DBpedia*, out of which 157 were found to be correct, giving a precision of 78.2%. Common patterns of alignments include alignments of an area identified by its *OpenGeoDb* location id with its name or license plate codes from *DBpedia*. We were not able to detect 14 alignments correctly, where there were multiple spellings for the same entity (e.g. *LinkedGeoData* uses both "Hof Oberfranken" and "Landkreis Hof Oberfranken" in its values for its *linkedgeodata:is_in* property). Another 20 alignments evaluated were partial (e.g. out of the 88 counties in Ohio, the algorithm produced a covering including only 54). There were some other errors as well (e.g. Places with license plate code GR in *DBpedia* were aligned with instances having license code GR, NOL & ZI in *LinkedGeoData*).

Linking Species from *Geospecies* with *DBpedia*: In aligning *Geospecies* with *DBpedia*, out of the 178 alignments that we evaluated, we found 109 correct alignments for a precision of 61.24%. For 25 of the results, due to the presence of multiple names/lexical values for the same item (e.g. both"Decapoda"@en and dbpedia:Decapoda values exist for *dbpedia:ordo* property). In 28 of the evaluated alignments, we were only able to find partial *concept coverings*, mostly because of insufficient instances and property assertions. For 16 other alignments, however,

Table 3. Linking *LinkedGeoData* with *DBpedia*

Description of Pattern Observed	Alignments w/ larger class from *LinkedGeoData*	Alignments w/ larger class from *DBpedia*
Total # Alignments	2746	3097
# Alignments Evaluated	**100**	**100**
Correct	**78**	**79**
Unidentified due to multiple spellings	5	9
Partially found	13	7
Other	4	5

Table 4. Linking *Geospecies* with *DBpedia*

Description of Pattern Observed	Alignments w/ larger class from *Geospecies*	Alignments w/ larger class from *DBpedia*
Total # Alignments	191	255
# Alignments Evaluated	**93**	**85**
Correct	**49**	**60**
Unidentified due to multiple spellings	25	0
Partially found	4	24
Other	15	1

there were some interesting reasons. In some cases, the biological classes were no longer in use (Urticales, Homoptera, etc.). There were some alignments that we were not able to guess correctly because the species were marked as belonging to different classification systems. There were also a few mismatches to a class at a different level in the hierarchy.

Linking Genes from *GeneID* with *MGI*: In the 28 alignments found between *GeneID* and *MGI*, 24 were found to be correct for a precision of 85.71%. Most (20) of these were alignments linking a gene start position from *MGI* with possible locations from *GeneID*. In theory, these are numeric distances in centimorgans and can actually be an infinite set. In the data however we find all possible distances occurring as text. The other 4 alignments were partial because of insufficient data.

Table 5. Linking *GeneID* with *MGI*

Description of Pattern Observed	Alignments w/ larger class from *GeneID*	Alignments w/ larger class from *MGI*
Total # Alignments	6	22
# Alignments Evaluated	**6**	**22**
Correct	**4**	**20**
Partially found	2	2

5 Related Work

Ontology alignment and schema matching have been a well explored area of research since the early days of ontologies [5,1] and received renewed interest in recent years with the rise of the Semantic Web and Linked Data. In the Web of Linked Data, even though most work done is on linking instances across different sources, an increasing number of authors have looked into aligning the source ontologies in the past couple of years. Jain et al. [8] describe the BLOOMS approach, which uses a central forest of concepts derived from topics in Wikipedia. An update to this is the BLOOMS+ approach [9] that aligns Linked Open Data ontologies with an upper-level ontology called Proton. BLOOMS is unable to find alignments because of the single *Feature* class in *GeoNames*. BLOOMS+, which uses contextual information, finds some alignments between *GeoNames* & Proton (precision of 0.5%) and *DBpedia* & Proton (90%). Cruz et al. [2] describe a dynamic ontology mapping approach called *AgreementMaker* that uses similarity measures along with a mediator ontology to find mappings using the labels of the classes. From the subset and equivalent alignment between *GeoNames* (10 concepts) and *DBpedia* (257 concepts), AgreementMaker achieves a precision of 26% and a recall of 68%. In comparison, for *GeoNames* and *DBpedia*, we achieve a precision of 64.4%. But this comparison does not reflect that we find concept coverings in addition to one-to-one alignments, while the other systems only find one-to-one alignments. The advantage of our approach over these is that our use of *restriction classes* is able to find a large set of alignments in cases like aligning *GeoNames* with *DBpedia* even in the presence of a rudimentary ontology. We believe that since other approaches do not consider concept descriptions beyond those in the original ontology (like *concept coverings*), they would not have been able to find alignments like the Educational Institutions example (#1) by using only the labels and the structure of the ontology.

Extensional techniques and concept coverings have also been studied in the past [7]. Völker et al. [13] describe an approach, similar to our work, that uses statistical methods for finding alignments. This work induces schemas for RDF data sources by generating OWL-2 axioms using an intermediate associativity table of instances and concepts (called *transaction datasets*) and mining associativity rules from it. The GLUE [3] system is a instance-based matching algorithm, which first predicts the concept in the other source that instances belong to using machine learning. GLUE then hypothesizes alignments based on the probability distributions obtained from the classifications. Our approach, in contrast, depends on the existing links (in Linked Open Data Cloud), and hence reflects the nature of the source alignments in practice. *CS*R [12] is a similar work to ours that tries to align a concept from one ontology to a union of concepts from the other ontology using the similarity of properties as features in predicting the subsumption relationships. It differs from our approach in that it uses a statistical machine learning approach for detection of subsets rather than the extensional approach.

Table 6. Example alignments from *GeoNames-DBpedia* *LinkedGeoData-DBpedia*

#	r_1	$p_2 = \{v_2\}$	$R'_U = \frac{\|U_A\|}{\|U_L\|}$	$\|U_A\|$	$\|U_L\|$	***Outliers***	***# Explained Instances***
***DBpedia* (larger) - *GeoNames* (smaller)**							
1	As described in Section 4, Schools, Colleges and Universities in *GeoNames* make Educational Institutions in *DBpedia*						
	rdf:type = dbpedia:EducationalInstitution	*geonames:featureCode* = {S.SCH, S.SCHC, S.UNIV}	0.9801	396	404	S.BLDG (3/122), S.EST (1/13), S.LIBR (1/7), S.HSP (1/31), S.MUS (1/43)	403
2	The concepts for the country Spain are equal in both sources. The only outlier has its country as Italy, an erroneous link.						
	dbpedia:country = dbpedia:Spain	*geonames:countryCode* = {*ES*}	0.9997	3917	3918	IT (1/7635)	3918
3	We confirm the hierarchical nature of administrative divisions with alignments between administrative units at two different levels.						
	dbpedia:region = dbpedia:Basse-Normandie	*geonames:parentADM2* = {geonames:2989247, geonames:2996268, geonames:3029094}	1.0	754	754		754
4	In aligning airports, an airfield should have been an an airport. However, there was not enough instance support.						
	rdf:type = dbpedia:Airport	*geonames:featureCode* = {S.AIRB, S.AIRP}	0.9924	1981	1996	S.AIRF (9/22), S.FRMT (1/5), S.SCH (1/404), S.STNB (2/5) S.STNM (1/36), T.HLL (1/61)	1996
***GeoNames* (larger) - *DBpedia* (smaller)**							
5	The Alignment for Netherlands should have been as straightforward as #2. However we have possible alias names, such as *The Netherlands* and *Kingdom of Netherlands*, as well a possible linkage error to *Flag_of_the_Netherlands.svg*						
	geonames:countryCode = NL	*dbpedia:country* = {dbpedia:The_Netherlands, dbpedia:Flag_of_the _Netherlands.svg, dbpedia:Netherlands}	0.9802	1939	1978	dbpedia:Kingdom_of _the_Netherlands (1/3)	1940
6	The error pattern in #5 seems to repeat systematically, as can be seen from this alignment for the country of Jordan.						
	geonames:countryCode = JO	*dbpedia:country* = {dbpedia:Jordan, dbpedia:Flag_of_Jordan.svg}	0.95	19	20		19
***DBpedia* (larger) - *LinkedGeoData* (smaller)**							
7	Our algorithm also produces interesting alignments between different properties. In this case, we find 8 of the 10 license plates in the state of Saarland. Instances supporting the remaining 2 license plates were missing						
	dbpedia:bundesland = Saarland	*lgd:OpenGeoDBLicensePlate-*Number = { HOM, IGB, MZG, NK, SB, SLS, VK, WND}	0.93	46	49		46

Table 7. Example alignments from *LinkedGeoData-DBpedia*, *Geospecies-DBpedia*

#	r_1	$p_2 = \{v_2\}$	$R'_U = \frac{\lvert U_A \rvert}{\lvert U_L \rvert}$	$\lvert U_A \rvert$	$\lvert U_L \rvert$	***Outliers***	***# Explained Instances***
8	Schools in *DBpedia* can be explained with types K2543 and Schools from *LinkedGeoData*						
	rdf:type= dbpedia:School	*rdf:type* = {lgd:School, lgd:K2543}	0.9907	2356	2378		2356
***LinkedGeoData* (larger) - *DBpedia* (smaller)**							
9	Due to missing instance links, this *concept covering* incorrectly claims that the state of New Jersey is composed of 9 counties while actually it has 21.						
	lgd:gnisST_alpha = NJ	*dbpedia:subdivisionName* = {Atlantic, Burlington, Cape May, Hudson, Hunterdon, Monmoth, New Jersey, Ocean, Passaic}	1.0	214	214		214
10	Waterways in *LinkedGeoData* is equal to the union of streams and rivers from *DBpedia*						
	rdf:type = lgd:Waterway	*rdf:type* = {dbpedia:River dbpedia:Stream}	0.97	33	34	dbpedia:Place(1/94989)	34
***DBpedia* (larger) - *Geospecies* (smaller)**							
11	Species from *Geospecies* with the order names Anura, Caudata & Gymnophionia are all Amphibians We also find inconsistencies due to misaligned instances, e.g. one amphibian was classified as a Turtle (Testudine) .						
	rdf:type = dbpedia:Amphibian	*geospecies:hasOrderName* = {Anura, Caudata, Gymnophionia}	0.99	90	91	Testudines (1/7)	91
12	Upon further inspection of #11, we find that the culprit is a Salamander						
	rdf:type = dbpedia:Salamander	*geospecies:hasOrderName* = {Caudata}	0.94	16	17	Testudines (1/7)	17
***Geospecies* (larger) - *DBpedia* (smaller)**							
13	The Kingdom Plantae, from both sources, almost matches perfectly. The only inconsistent instance happens to be a fungus.						
	rdf:type = dbpedia:Plant	*geospecies:inKingdom* = {geospecies:kingdoms/Ab}	0.99	1874	1876	geospecies:kingdoms/Ac (1/8)	1875

Table 8. Example alignments from *Geospecies*-*DBpedia*and *GeneID*-*MGI*

#	r_1	$p_2 = \{v_2\}$	$R'_U = \frac{\lvert U_A \rvert}{\lvert U_L \rvert}$	$\lvert U_A \rvert$	$\lvert U_L \rvert$	***Outliers***	***# Explained Instances***
14	Inconsistencies in the object values can also be seen - Carnivores from *Geospecies* are aligned with both Carnivora and Carnivore.						
	geospecies:inOrder = geospecies:orders/jtSaY	*dbpedia:ordo =* {dbpedia:Carnivora, dbpedia:Carnivore}	0.99	246	247		246
15	We can detect that species with order Chiroptera correctly belong to the order of Bats. Unfortunately, due to values of the property being the literal "Chiropta@en", the alignment is not clean.						
	geospecies:hasOrderName = Chiroptera	*dbpedia:ordo =* {Chiroptera@en, dbpedia:Bat}	1	111	111		111
***GeneID* (larger) - *MGI* (smaller)**							
16	The classes for Pseudogenes align.						
	bio2rdf:subType = pseudo	*bio2rdf:subType =* {Pseudogene}	0.93	5919	6317	Gene (318/24692)	6237
17	The Mus Musculus (house mouse) genome is composed of complex clusters, DNA segments, Genes and Pseudogenes.						
	bio2rdf:xTaxon = taxon:10090	*bio2rdf:subType =* {Complex Cluster/Region, DNA Segment, Gene, Pseudogene}	1	30993	30993		30993
***MGI* (larger) - *GeneID* (smaller)**							
18	Inconsistencies are also evident as the values pseudo and Pseudogene are used to denote the same thing.						
	bio2rdf:subType = Pseudogene	*bio2rdf:subType = {pseudo}*	0.94	5919	6297	other (4/230) protein-coding (351/39999) unknown(23/570)	6297
19	We find alignments like #19 & #20 , which align the gene start (with the chromosome) in *MGI* with the location in *GeneID* As can be seen, the values of the locations (distances in centimorgans) in *GeneID* contain the chromosome as a prefix. Inconsistencies are also seen, e.g. in #19 a gene that starts with 5 is misaligned and in #20, where the value is an empty string.						
	mgi:genomeStart = 1	*geneid:location =* {1, 1 0.0 cM, 1 1.0 cM, 1 10.4 cM, ...}	0.98	1697	1735	""(37/1048) 5 (1/52)	1735
20	*mgi:genomeStart = X*	*geneid:location =* {X, X 0.5 cM, X 0.8 cM, X 1.0 cM, ...}	0.99	1748	1758	""(10/1048)	1758

6 Conclusions and Future Work

We described an approach to identifying *concept coverings* in Linked Data sources from the Geospatial, Biological Classification and Genetics domains. By introducing the definition of *restriction classes* with the disjunction operator, we are able to find alignments of union concepts from one source to larger concepts from the other source. Our approach produces coverings where concepts at different levels in the ontologies of two sources can be mapped even when there is no direct equivalence or only rudimentary ontologies exist. Our algorithm is also able to find outliers that help identify erroneous links or inconsistencies in the linked instances. Our results provide a deeper insight into the nature of the alignments of Linked Data.

In future work we want to find more complete descriptions for the sources. Our preliminary findings show that the results of this paper can be used to find patterns in the properties. For example, the *countryCode* property in *GeoNames* is closely associated with the *country* property in *DBpedia*, though their ranges are not exactly equal. By mining rules from the generated alignments, we will be closer to the interoperability vision of the Semantic Web. A second direction of future work is to use the outliers to feed the corrections back to the sources, particularly *DBpedia*, and to the RDF data quality watchdog group pedantic-web.org. To achieve this satisfactorily, we not only need to point out the instances that have errors, but suggest why those errors occurred, that is, whether it was due to incorrect assertions or missing links.

Acknowledgements. This research is based upon work supported in part by the National Science Foundation under award number IIS-1117913. 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. Bernstein, P., Madhavan, J., Rahm, E.: Generic schema matching, ten years later. Proceedings of the VLDB Endowment 4(11) (2011)
2. Cruz, I., Palmonari, M., Caimi, F., Stroe, C.: Towards on the go matching of linked open data ontologies. In: Workshop on Discovering Meaning on the Go in Large Heterogeneous Data, p. 37 (2011)
3. Doan, A., Madhavan, J., Domingos, P., Halevy, A.: Ontology matching: A machine learning approach. In: Handbook on Ontologies, pp. 385–404 (2004)
4. Duckham, M., Worboys, M.: An algebraic approach to automated geospatial information fusion. International Journal of Geographical Information Science 19(5), 537–558 (2005)
5. Euzenat, J., Shvaiko, P.: Ontology matching. Springer (2007)
6. Horrocks, I., Patel-Schneider, P., Van Harmelen, F.: From shiq and rdf to owl: The making of a web ontology language. Web Semantics: Science, Services and Agents on the World Wide Web 1(1), 7–26 (2003)

7. Isaac, A., van der Meij, L., Schlobach, S., Wang, S.: An Empirical Study of Instance-Based Ontology Matching. 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.) ISWC/ASWC 2007. LNCS, vol. 4825, pp. 253–266. Springer, Heidelberg (2007)
8. Jain, P., Hitzler, P., Sheth, A.P., Verma, K., Yeh, P.Z.: Ontology Alignment for Linked Open Data. 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. 402–417. Springer, Heidelberg (2010)
9. Jain, P., Yeh, P.Z., Verma, K., Vasquez, R.G., Damova, M., Hitzler, P., Sheth, A.P.: Contextual Ontology Alignment of LOD with an Upper Ontology: A Case Study with Proton. 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. 80–92. Springer, Heidelberg (2011)
10. Parundekar, R., Knoblock, C.A., Ambite, J.L.: Linking and Building Ontologies of Linked Data. 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. 598–614. Springer, Heidelberg (2010)
11. Parundekar, R., Knoblock, C.A., Ambite, J.L.: Finding concept coverings in aligning ontologies of linked data. In: Proceedings of the First International Workshop on Knowledge Discovery and Data Mining Meets Linked Open Data in Conjunction with the 9th Extended Semantic Web Conference, Heraklion, Greece (2012)
12. Spiliopoulos, V., Valarakos, A.G., Vouros, G.A.: *CSR*: Discovering Subsumption Relations for the Alignment of Ontologies. In: Bechhofer, S., Hauswirth, M., Hoffmann, J., Koubarakis, M. (eds.) ESWC 2008. LNCS, vol. 5021, pp. 418–431. Springer, Heidelberg (2008)
13. 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)

Ontology Constraints in Incomplete and Complete Data

Peter F. Patel-Schneider and Enrico Franconi

Free University of Bozen-Bolzano, Italy
pfpschneider@gmail.com, franconi@inf.unbz.it

Abstract. Ontology and other logical languages are built around the idea that axioms enable the inference of new facts about the available data. In some circumstances, however, the data is meant to be complete in certain ways, and deducing new facts may be undesirable. Previous approaches to this issue have relied on syntactically specifying certain axioms as constraints or adding in new constructs for constraints, and providing a different or extended meaning for constraints that reduces or eliminates their ability to infer new facts without requiring the data to be complete. We propose to instead directly state that the extension of certain concepts and roles are complete by making them DBox predicates, which eliminates the distinction between regular axioms and constraints for these concepts and roles. This proposal eliminates the need for special semantics and avoids problems of previous proposals.

1 Introduction

A complaint against ontology languages like the W3C OWL Web Ontology Language is that axioms in such ontology languages sometimes have too many consequences. For example,[1] the axiom $\mathsf{MarriedPerson} \sqsubseteq \forall \mathsf{hasSpouse}.\mathsf{Person}$ along with $\mathsf{MarriedPerson}(\mathsf{Peter})$ and $\mathsf{hasSpouse}(\mathsf{Peter}, \mathsf{Susan})$ produces the consequence $\mathsf{Person}(\mathsf{Susan})$. Similarly, the axiom $\mathsf{MarriedPerson} \equiv \mathsf{Person} \sqcap (=1\, \mathsf{hasSpouse})$ along with $\mathsf{MarriedPerson}(\mathsf{Peter})$ implies that there is some spouse for Peter. However, the data generation methodology may be such that all persons and/or all spouse relationships should be explicitly given, and not inferred. For example, the data for these concepts or roles may come from an ostensibly complete database. If this is the case, then problems arise when using ontology axioms in the usual way. In the first example Susan is inferred to be a person even if there may be no explicit fact to this effect (and thus Susan is not a person) and in the second example some (unknown) spouse is inferred for Peter, which is not an explicit fact at all.

This particular complaint against standard ontology languages has spawned a number of proposals attempting to overcome the problem. Some proposals [10,15] change certain axioms into integrity constraints (roughly statements that

[1] Throughout this paper a standard concise syntax [1] for OWL and other ontology languages and description logics will be used in examples to improve readability.

check the integrity of the data in the knowledge base (KB) instead of enabling consequences) that are interpreted in minimal or modal ways. Other proposals [6] suggest that the ontology language be extended with autoepistemic constructs that can be used to write integrity constraints.

We disagree with the general approach taken in all of these proposals. Our belief is that the problem here is not a deficiency in ontology languages at all. In particular, we do not find any problems with the ontology axioms above, nor with their consequences. To the contrary, these axioms are unexceptional and should infer *consequences* just like any other axiom. Instead we claim that the complaints against the power of axioms in ontology languages have to do with a mismatch between the general open-world assumption in ontology languages and in many other logics, on one hand, and a desire to have complete and maybe even explicit information about the extension of certain concepts and roles, on the other.

So, for example, if we have complete *data* about some concepts or roles, e.g., a set of the distinct instances of Person or the distinct pairs in hasSpouse, then it should be the case that inferences *do not* augment this data. But, again, this is not a problem with the axioms, e.g., the ones above, which are simply a reflection of the way we believe the domain is, but is instead a problem with our data, which is incomplete when it was stated to be complete. A consequence that adds more information to these concepts or roles then is contradicting this supposed completeness, causing an inconsistency in the KB.

Our basic approach to modeling, then, is to build the ontology without considering anything like the desired completeness of data or integrity constraints. Only then do we determine which concepts and roles are to have complete data. So, for example, we might build the ontology of people and spouses

$$\mathsf{Person} \sqsubseteq \top$$
$$\mathsf{MarriedPerson} \equiv \mathsf{Person} \sqcap (= 1\,\mathsf{hasSpouse})$$
$$\mathsf{MarriedPerson} \sqsubseteq \forall \mathsf{hasSpouse}.\mathsf{Person}$$

We might, then, perhaps because we are using an explicit data source of all spousal relationships, require that the hasSpouse role be completely and explicitly provided.

Suppose that data in the hasSpouse role is provided as the four distinct entries

hasSpouse

Peter	Susan
Susan	Peter
Mary	Paul
Paul	Mary

(1)

Our ontology axioms will make inferences from this data, including Person(Susan) and MarriedPerson(Peter) which is all as it should be. Suppose, however, that our KB also includes MarriedPerson(Alex) Then our ontology axioms would, in effect, add another entry to the hasSpouse role, one with first entry Alex. As we stated that the given data for hasSpouse is complete, this would be an inconsistent situation indicating that the constraint has been violated.

This treatment of complete concepts and roles is just the DBox treatment [14,7]. Our approach has all the benefits of DBoxes, including easy query answering when all concepts and roles are DBox concepts and roles, and *exact answers* to queries controlled by DBox concepts and roles, i.e., if we query for married people and their spouses above we get precisely every spousal relationship. These exact answers can be used in applications without worrying that there might be missing information, i.e., as might happen when there is a married person whose spouse is not known. Applications can treat such query answers in the same way that they can treat query answers in databases.

2 Autoepistemic and Minimal Model Approaches

Autoepistemic extensions to description logics or ontology languages, such as the work by Donini *et al* on description logics of minimal knowledge and negation as failure (MKNF-DL) [6], can be used to express constraints, either directly, or as rules [12]. Negation as failure by itself has been used to express constraints in OWL Flight [5,4].

The MKNF-DL axiom

$$\mathbf{K}\mathsf{MarriedPerson} \sqsubseteq \exists\mathbf{A}\mathsf{hasSpouse} \tag{2}$$

expresses the constraint that every known married person has a known spouse so that a KB containing only

$$\begin{array}{l} \mathbf{K}\mathsf{MarriedPerson} \sqsubseteq \exists\mathbf{A}\mathsf{hasSpouse} \\ \mathsf{MarriedPerson}(\mathsf{Joe}) \end{array} \tag{2.1}$$

is unsatisfiable because Joe does not have a known spouse. Knowledge in MKNF-DL requires knowing what, not just knowing that, so

$$\begin{array}{ll} \mathbf{K}\mathsf{MarriedPerson} \sqsubseteq \exists\mathbf{A}\mathsf{hasSpouse} & \\ \mathsf{MarriedPerson}(\mathsf{Joe}) & (\exists\mathsf{hasSpouse})(\mathsf{Joe}) \end{array} \tag{2.2}$$

is also unsatisfiable because, although Joe does have some spouse, the identity of that spouse is not known. Only information about the identity of Joe's spouse is adequate to satisfy the constraint, as in

$$\begin{array}{ll} \mathbf{K}\mathsf{MarriedPerson} \sqsubseteq \exists\mathbf{A}\mathsf{hasSpouse} & \\ \mathsf{MarriedPerson}(\mathsf{Joe}) & \mathsf{hasSpouse}(\mathsf{Joe}, \mathsf{Susan}) \end{array} \tag{2.3}$$

Adding autoepistemic constructs into description logics augments their expressive power considerably. Although the result is decidable, we are unaware of any high-performance systems that implement reasoning in MKNF-DL.

The constraint axioms in the MKNF-DL approach involve autoepistemic operators in negative contexts, such as $\mathbf{K}\mathsf{MarriedPerson}$ above, indicating that lack of knowledge can allow a constraint to be satisfied. Constraint axioms thus do not necessarily imply their non-constraint version, so one often needs to provide two versions of constraint axioms, such as by adding

$$\mathsf{MarriedPerson} \sqsubseteq {=}1\ \mathsf{hasSpouse}$$

Without such an axiom, as in KB 2.1, it will not be the case that all married-Persons have a spouse, only the known ones.

For similar reasons, the MKNF-DL approach does not require that existential individuals be considered in constraints. For example, in

$$\begin{aligned}&\mathbf{K}\mathsf{MarriedPerson} \sqsubseteq \exists\mathbf{A}\mathsf{hasSpouse} \\ &(\exists\mathsf{hasChild.MarriedPerson})(\mathsf{Mary})\end{aligned} \tag{2.4}$$

the constraint is satisfied even though nothing is known about Mary's married child. We feel that constraints should be active on all participating individuals, independently of whether they are known, and consider that the example above points out a major problem with MKNF-DL and similar modal approaches.

Also similarly, the MKNF-DL approach does not require that disjunctive information be considered by constraints. So

$$\begin{aligned}&\mathsf{Student} \equiv \mathsf{UGStudent} \sqcup \mathsf{GStudent} \\ &\mathbf{K}\mathsf{UGStudent} \sqsubseteq \exists\mathbf{A}\mathsf{major} \\ &\mathbf{K}\mathsf{GStudent} \sqsubseteq \exists\mathbf{A}\mathsf{faculty} \\ &\mathsf{Student}(\mathsf{Mary})\end{aligned} \tag{3}$$

is satisfiable even though Mary must be either an undergraduate, in which case the constraint about majors is not satisfied, or a graduate, in which the constraint about faculties is not satisfied. We feel that constraints should take into account such disjunctive information and consider that this example points out another major problem with this kind of approach.

Motik *et al* [10] have proposed a very different approach to integrity constraints. Instead of extending the language itself, they divide up axioms into three categories. As is usual, facts are segregated into the ABox ($\mathcal{A}$). Other axioms, however, are divided between regular axioms (the standard TBox or $\mathcal{S}$) and constraints ($\mathcal{C}$). Next, minimal Herbrand models are defined. In this approach a minimal Herbrand model is a Herbrand model where the extension of all predicates, even equivalence (the predicate standing in for equality), is minimized. A constraint is then satisfied by a standard TBox and an ABox if all minimal Herbrand models of the standard TBox plus the ABox are also models of the constraint. Because each minimal model of the standard TBox plus the ABox is a model of the standard TBox, it is obvious that in this approach any constraint entailed by the standard TBox is satisfied in the KB.

In the KB

$$\begin{aligned}&\mathcal{C} : \mathsf{MarriedPerson} \sqsubseteq \exists\mathsf{hasSpouse} \\ &\mathcal{A} : \mathsf{MarriedPerson}(\mathsf{Joe})\end{aligned} \tag{2.1_m}$$

the first axiom is considered as a constraint and the second as a fact in the ABox. The constraint is not satisfied in this KB because in the (only) minimal model of $\mathcal{A}$ Joe has no spouse.

The minimal model approach differs considerably from the MKNF-DL approach. For example in

$$\begin{aligned}&\mathcal{S} : \mathsf{MarriedPerson} \sqsubseteq \exists \mathsf{hasSpouse}\\ &\mathcal{C} : \mathsf{MarriedPerson} \sqsubseteq \exists \mathsf{hasSpouse}\\ &\mathcal{A} : \mathsf{MarriedPerson(Joe)}\end{aligned} \qquad (2.2_m)$$

the constraint is satisfied, even though there is no known spouse for Joe, because in each minimal model, Joe has a spouse. We consider this ability to utilize "unknown", or existential, fillers to fulfill constraints as a problem with this approach.

The particular problem with existentials in the previous example can be alleviated by introducing an extra predicate (O) that is asserted true of every identifier in the KB, as in

$$\begin{aligned}&\mathcal{S} : \mathsf{MarriedPerson} \sqsubseteq \exists \mathsf{hasSpouse}\\ &\mathcal{C} : \mathsf{MarriedPerson} \sqsubseteq \exists \mathsf{hasSpouse} \sqcap \mathsf{O}\\ &\mathcal{A} : \mathsf{MarriedPerson(Joe)} \qquad \mathsf{O(Joe)}\end{aligned} \qquad (2.2\text{'}_m)$$

Here Joe's spouse is not in O so the constraint is not satisfied. One can think of O as holding the requirement of having a name.

However, other problems with this minimal model approach cannot be overcome. If a filler can be one of two possibilities (both named) then a constraint requiring a filler will be satisfied, even though the identity of the filler is not known. For example, in

$$\begin{aligned}&\mathcal{S} : \mathsf{JoeClass} \sqsubseteq \exists \mathsf{hasSpouse}.\{\mathsf{Mary}, \mathsf{Susan}\}\\ &\mathcal{C} : \mathsf{MarriedPerson} \sqsubseteq \exists \mathsf{hasSpouse}\\ &\mathcal{A} : \mathsf{MarriedPerson(Joe)} \qquad \mathsf{JoeClass(Joe)}\end{aligned} \qquad (2.5_m)$$

the constraint is satisfied even though Joe's spouse is not known. We consider this to be a major problem with this approach, as we view the goal of integrity constraints to be checking data, not checking possibilities.

Another problem with the minimal models approach is that all concepts and roles are minimized. This means that the presence of axioms that cause one concept or role to grow when something else shrinks disturbs the minimization in unusual ways. For example, the constraint in

$$\begin{aligned}&\mathcal{C} : \mathsf{RParent} \sqsubseteq \geqslant 2\, \mathsf{hasChild}\\ &\mathcal{A} : \mathsf{RParent(Joe)} \qquad \mathsf{hasChild(Joe, Mary)} \qquad \mathsf{hasChild(Joe, Susan)}\end{aligned} \qquad (4)$$

is satisfied, as expected, because the minimal model minimizes away the possible equality between Mary and Susan. However, extending the KB by adding a definition, as in

$$\begin{aligned}&\mathcal{S} : \mathsf{DParent} \equiv \geqslant 2\, \mathsf{hasChild}\\ &\mathcal{C} : \mathsf{RParent} \sqsubseteq \geqslant 2\, \mathsf{hasChild} \qquad (4.1)\\ &\mathcal{A} : \mathsf{RParent(Joe)} \quad \mathsf{hasChild(Joe, Mary)} \quad \mathsf{hasChild(Joe, Susan)}\end{aligned}$$

can make the constraint not be satisfied. The reason for this *unexpected* result is that DParent grows when equality shrinks. This results in a minimal model where Mary and Susan are the same and Joe is not in DParent. In this minimal model Joe has only one child and this violates the constraint. We view this as a very serious problem with the approach, particularly as facts and queries in the approach have to use atomic concepts and roles, and thus may require the introduction of extra predicates.

A major reason for the difference between the MKNF-DL approach and the minimal model approach has to do with the modal (or non-modal) nature of the approaches. The modal parts of constraints in MKNF-DL consider what is known to be true in all models, so **K**UGStudent picks out named individuals who are known to be graduate students in all models. In the minimal model approach constraints are directly evaluated in each minimal model, so there is no consideration of the situation in other models.

A third approach [15] to integrity constraints combines axiom segregation and equality minimization, somewhat as in the minimal model approach, with a portion of the autoepistemic nature of the MKNF-DL approach. In this hybrid approach, there is a two-way division in extended KBs between $\mathcal{K}$, the regular KB ($\mathcal{T}$ and $\mathcal{A}$ of the previous approach), and $\mathcal{S}$, the constraints, both of which can contain both axioms and facts. The minimal equality models of a KB are defined as the regular models of the KB that are minimal with respect to equality between individuals names, with all else remaining fixed. Constraints are interpreted in a modal setting where atomic concepts and roles are interpreted, roughly, as names (pairs of names) belonging to the concept (role) in all minimal models of the KB.

One might think that only minimizing equality avoids the problems in the previous approach with respect to additional predicates. Unfortunately, this is not the case. For example, the extended KB

$$\begin{aligned}&\mathcal{C} : \mathsf{RParent} \sqsubseteq \geqslant 2\, \mathsf{hasChild} \qquad (4_h)\\ &\mathcal{T} : \mathsf{RParent(Joe)} \quad \mathsf{hasChild(Joe, Mary)} \quad \mathsf{hasChild(Joe, Susan)}\end{aligned}$$

is valid, as expected, but the extended extended KB

$$\begin{aligned}&\mathcal{C} : \mathsf{RParent} \sqsubseteq \geqslant 2\, \mathsf{hasChild}\\ &\mathcal{T} : \mathsf{DParent} \equiv \geqslant 2\, \mathsf{hasChild} \qquad (4.1_h)\\ &\quad\ \mathsf{RParent(Joe)} \quad \mathsf{hasChild(Joe, Mary)} \quad \mathsf{hasChild(Joe, Susan)}\end{aligned}$$

is *not*. Here the presence of DParent prevents minimizing away the possible equality between Mary and Susan, because making Mary and Susan the same causes a change in the extension of DParent. As is the case for the previous approach, we view this fragility of minimization as a very serious problem.

This combination approach also suffers from variants of many of the problems of the MKNF-DB approach, including both extended KBs

$$\begin{aligned}&\mathcal{C} : \mathsf{Child} \sqsubseteq \bot \\ &\mathcal{T} : (\exists \mathsf{hasSpouse}.\mathsf{Spouse})(\mathsf{Joe})\end{aligned} \tag{5.1}$$

and

$$\begin{aligned}&\mathcal{C} : \mathsf{Child} \sqsubseteq \bot \\ &\mathcal{T} : (\forall \mathsf{hasSpouse}.\mathsf{Child})(\mathsf{Joe}) \qquad (\exists \mathsf{hasSpouse}.\{\mathsf{Mary}, \mathsf{Susan}\})(\mathsf{Joe})\end{aligned} \tag{5.2}$$

being valid. However, the situation is even worse here, as one might argue that the MKNF-DB construct analogous to the constraint, $\mathsf{Child} \sqsubseteq \bot$, should not be considered to be a constraint, but such arguments cannot be made when constraints are explicitly given.

Local closed world semantics has a relationship to integrity constraints, as can be seen from the fact that all of the above approaches employ some form of minimization. A proposal to add local closed world semantics to OWL [13] uses grounded circumscription to avoid undecidability problems with circumscription. Grounded circumscription is just regular circumscription, except that minimized concepts (roles) can only contain named individuals (pairs of named individuals).

Grounded circumscription can capture some common aspects of integrity constraints. The basic idea is to evaluate the constraints after grounded circumscription has been applied. The advantage of circumscription over other minimization methods is that only certain predicates are minimized, while other are fixed or allowed to vary. In this way it might be possible to alleviate (but probably not completely overcome) the problems of simpler minimization methods.

A major problem with this approach, however, is the difficulty of performing even grounded circumscription. When circumscribing, one has to guess which named individuals (pairs of named individuals) are in each minimized concept (role), and only then determine whether the guess is acceptable. Then the minimal acceptable guesses become the actual minimizations. When the KB is even of only a moderate size, this can take an extremely long time.

3 Constraints with Complete Information

Our proposal does not depend on any of these modal or minimal model techniques to prevent constraints from enabling inferences. Instead, as stated earlier, we specify that certain concepts and roles have complete information. Then for these concepts and roles no information can be added, turning axioms into constraints for them. In effect, axioms can only check that information is already in the complete concept or role, precisely as is wanted for integrity constraints.

If an axiom plus some data produces a consequence that adds information to a complete concept or role, then an inconsistency results. What happens after an inconsistency is detected is outside the scope of the logic. Generally, some modification would be needed to the assertions in the KB, which might involve removing an assertion that enabled the attempt to add the offending information. The modification might, on the other hand, actually be to change the information in the complete concept or role, but this would occur as a step outside of the logic.

3.1 OWL and $\mathcal{SROIQ}(\mathcal{D})$

As much of our proposal is related to the W3C OWL Web Ontology Language [11] and related ontology languages, we introduce OWL, via $\mathcal{SROIQ}(\mathcal{D})$ [9], the description logic underlying OWL.

Let $\mathbf{C}$ be a set of *concept names*, $\mathbf{D}$ be a set of *datatype names*, $\mathbf{R}$ be a set of *abstract role names*, $\mathbf{T}$ be a set of *concrete role names*, $\mathbf{I}$ be a set of *individual names*, and $\mathbf{V}$ be a set of *data values*, with $\mathbf{C} \cap \mathbf{D} = \phi$, $\mathbf{R} \cap \mathbf{T} = \phi$, and $\mathbf{I} \cap \mathbf{V} = \phi$.

$\mathcal{SROIQ}(\mathcal{D})$ concepts (C), datatypes (D), and abstract roles (R) are constructed via

$$\begin{aligned}
C &::= \top \mid A \mid \{a\} \mid \neg C \mid C_1 \sqcap C_2 \mid C_1 \sqcup C_2 \mid \exists R.C \mid \forall R.D \mid \exists R.Self \mid \\
&\qquad \geqslant n\, R.C \mid \leqslant n\, R.C \mid \exists T.D \mid \forall T.D \mid \geqslant n\, T.D \mid \leqslant n\, T.D \\
D &::= B \mid \{v\} \mid \neg D \mid D_1 \sqcap D_2 \mid B \leq v \mid B < v \mid B > v \mid B \geq v \\
R &::= \top \mid P \mid P^{-}
\end{aligned}$$

where $A \in \mathbf{C}$, $B \in \mathbf{D}$, $P \in \mathbf{R}$, $T \in \mathbf{T}$, $a \in \mathbf{I}$, $v \in \mathbf{V}$, and n is a non-negative integer.

A general concept inclusion axiom (GCI) is $C_1 \sqsubseteq C_2$, for C_1, C_2 both concepts. A role inclusion axiom (RIA) is $R_1 \circ \ldots \circ R_n \sqsubseteq R$, for R, R_i all roles or $T_1 \sqsubseteq T$, for $T, T_1 \in \mathbf{T}$.

A role assertion is *Sym*(R), *Tra*(R), *Ref*(R), *Irr*(R), or *Dis*(R_1, R_2), for R, R_1, R_2 any role except $\top$; or $\mathsf{Dis}(T_1, T_2)$, for $T_1, T_2 \in \mathbf{T}$. An individual assertion is $C(a)$, $R(a_1, a_2)$, $(\neg R)(a_1, a_2)$, $T(a, v)$, $a_1 = a_2$, or $a_1 \neq a_2$, for C a concept, R a role, $T \in \mathbf{T}$, $a, a_1, a_2 \in \mathbf{I}$, and $v \in \mathbf{V}$.

A $\mathcal{SROIQ}(\mathcal{D})$ KB is a pair $\langle \mathcal{T}, \mathcal{A} \rangle$. $\mathcal{T}$ (the TBox) is a finite set of GCIs and RIAs and role assertions such that the RIAs in $\mathcal{T}$ form a regular role hierarchy (see [9]), the role assertions in $\mathcal{T}$ are simple in $\mathcal{T}$, and each role in an $\geqslant R C$. or $\exists R.Self$ or $\mathsf{Irr}(R)$ or $\mathsf{Dis}(R_1, R_2)$ is simple in $\mathcal{T}$ (see [9]). $\mathcal{A}$ (the ABox) is a finite set of individual assertions such that each role, R, in a $(\neg R)(a, b)$ is simple in $\mathcal{T}$. The names and values of the KB form its signature, $\mathcal{S}$.

The semantics of $\mathcal{SROIQ}(\mathcal{D})$ are as for $\mathcal{SROIQ}$ [9] and as for the W3C OWL 2 Web Ontology Language [11]. Here we present only the general notions of their semantics. Much of our proposal works for any ontology language or description logic or even any fragment of first-order logic, so we will provide a a general semantic framework that is suitable for any of these languages.

Semantics are based on interpretations, $\mathcal{I}$, that map, $\cdot^{\mathcal{I}}$, constants (individual names and data values) into elements of a domain, $\Delta^{\mathcal{I}}$, concept and datatype names into subsets of the domain, and abstract and concrete role names into sets of pairs over the domain. Data values and datatype names have fixed mapping.

This mapping is extended to all syntactic constructs in the language, mapping closed formulae (axioms and assertions) into either true or false (satisfying or not satisfying them, respectively). An interpretation is a model of a KB consisting of a finite set of closed formulae (axioms and assertions), written $\mathcal{I} \models \mathcal{KB}$, iff it satisfies all the formulae the KB.

3.2 DBoxes

Given a $\mathcal{SROIQ}(\mathcal{D})$ (or other description logic) KB, a DBox [14], $\mathcal{DB}$, is a finite set of atomic individual assertions of the form $A(a)$ or $P(a_1, a_2)$ or $T(a, v)$ for $A \in \mathbf{C}$, $P \in \mathbf{R}$, $T \in \mathbf{T}$, $a, a_1, a_2 \in \mathbf{I}$, and $v \in \mathbf{V}$. The signature of a DBox $\mathcal{DB}$, called $\mathcal{S}(\mathcal{DB})$, contains all the concept, abstract or concrete role names occurring in $\mathcal{DB}$. A $\mathcal{SROIQ}(\mathcal{D})$ (or other description logic) KB plus DBox is a triple $\langle \mathcal{T}, \mathcal{A}, \mathcal{DB} \rangle$ where $\mathcal{T}$ and $\mathcal{A}$ are as before and $\mathcal{DB}$ is a DBox. The active domain of a DBox $\mathcal{DB}$, $\mathbf{I}_{\mathcal{DB}} \subseteq \mathbf{I}$, is the set of all individuals appearing in the DBox. An interpretation, $\mathcal{I}$, of $\langle \mathcal{T}, \mathcal{A}, \mathcal{DB} \rangle$ is just an interpretation of $\langle \mathcal{T}, \mathcal{A} \rangle$ plus

- for each individual name $a \in \mathbf{I}_{\mathcal{DB}}$: $a^{\mathcal{I}} = a$, (i.e., the standard name assumption for DBox individuals);
- for each pair of distinct individual names $a_1 \in \mathbf{I}_{\mathcal{DB}}$ and $a_2 \in \mathbf{I}$: $a_1^{\mathcal{I}} \neq a_2^{\mathcal{I}}$, (i.e., the unique name conditions for DBox individuals);
- for each concept name $A \in \mathcal{S}(\mathcal{DB})$, $\quad x \in A^{\mathcal{I}}$ iff $\exists A(a) \in \mathcal{DB} \ : \ a^{\mathcal{I}} = x$;
- for each abstract role name, $P \in \mathcal{S}(\mathcal{DB})$,
  $\langle x, y \rangle \in P^{\mathcal{I}}$ iff $\exists P(a_1, a_2) \in \mathcal{DB} \ : \ a_1^{\mathcal{I}} = x \wedge a_2^{\mathcal{I}} = y$; and
- for each concrete role name, $T \in \mathcal{S}(\mathcal{DB})$,
  $\langle x, y \rangle \in T^{\mathcal{I}}$ iff $\exists T(a, v) \in \mathcal{DB} \ : \ a^{\mathcal{I}} = x \wedge v^{\mathcal{I}} = y$.

The essence of a DBox is that the extension of each concept or role that shows up in the DBox is completely determined by the DBox, much as it would be by a database table. This requires the standard name assumption in the DBox. To emphasize the relationship between database tables and DBox concept and roles, we will often write the assertions for the concept or role in tabular form, as in the case of the hasSpouse role in the KB (1) above.

It can been shown (see [8]) that it is harmless to drop the standard name assumption for DBox individuals, in presence of the unique name conditions for DBox individuals: the spurious models in the weaker KB are indistinguishable from the good ones. It is also possible to fully encode a KB with a DBox into an equivalent KB in an expressive description logic such as $\mathcal{SROIQ}(\mathcal{D})$ or OWL (see [7]). The unique name conditions for DBox individuals can be easily written as a finite set of individual inequality assertions. To rewrite a DBox concept, C, with $C(i_1), \ldots, C(i_n)$ in the DBox, simply add the DBox assertions for C to the ABox and add $C \sqsubseteq \{i_1, \ldots, i_n\}$ to the TBox. To rewrite a DBox abstract or concrete role, R, with $R(i_1, v_1), \ldots, R(i_n, v_n)$ in the DBox, simply add the DBox assertions for R to the ABox, add $\exists R \sqsubseteq \{i_1, \ldots, i_n\}$ to the TBox, and, for each i_j, $1 \leq j \leq n$, add $(\forall R.\{v_{j_1}, \ldots, v_{j_{m_k}}\})(i_j)$ to the ABox, where $\{v_{j_1}, \ldots, v_{j_{m_k}}\}$ is the set of R-fillers for i_j in the DBox. As a consequence of this easy polynomial embedding, we can conclude that reasoning with DBoxes in such expressive ontology languages is not harder than classical reasoning without DBoxes, and it can be implemented without changing anything with respect to the classical case.

3.3 Completely Specified Concepts and Roles

We introduce in this section the definition of concepts and roles *completely specified* from DBox predicates, a notion strictly related to *determinacy* and *implicit definability* [2,8].

Definition (Completely Specified Concept or Role). *Let $\mathcal{I}$ and $\mathcal{J}$ be any two models of a KB plus DBox $\langle \mathcal{T}, \mathcal{A}, \mathcal{DB} \rangle$. A concept (resp. role) C (resp. R) is* completely specified *(or* determined*) by the DBox predicates $\mathcal{S}(\mathcal{DB})$ in the KB if and only if whenever I and J agree on the interpretation given to each concept and role in $\mathcal{S}(\mathcal{DB})$ then $C^{\mathcal{I}} = C^{J}$ (resp. $R^{\mathcal{I}} = R^{J}$).*

This definition states that a concept C (resp., a role R) is completely specified by the DBox predicates in a KB if and only if all models of the KB that interpret the symbols in $\mathcal{S}_{\mathbf{DB}}$ the same way also keep the interpretation for C (resp., R) fixed. In other words, once a DBox is fixed (and therefore the interpretation of all the DBox predicates is always the same in any interpretation) then also the interpretation of the completely specified predicates (concepts and/or roles) is fixed. It is as if the concept or role augments the original DBox with its own extension.

It is obvious that a DBox concept or role is completely specified by the DBox predicates, given the close correspondence between the definition of DBoxes and complete specification. It is also possible to completely specify a concept or role in other ways. For example, a concept or role might be defined to be equivalent to a DBox concept or role. Other, more complex, definitions can also completely specify a concept or role in terms of DBox concepts or roles. For example, from KBs of the following form it is possible to derive that $\mathsf{GStudent}$ is a completely specified concept, given that the concepts $\mathsf{Student}$ and $\mathsf{UGStudent}$ are DBox concepts:

$$\begin{aligned} \mathcal{T} :\ & \mathsf{Student} \sqsubseteq \mathsf{UGStudent} \sqcup \mathsf{GStudent} \\ & \mathsf{UGStudent} \sqsubseteq \neg\mathsf{GStudent} \sqcap \mathsf{Student} \\ & \mathsf{GStudent} \sqsubseteq \mathsf{Student} \\ \mathcal{S}(\mathcal{DB}) :\ & \{\mathsf{Student}, \mathsf{UGStudent}\} \end{aligned} \tag{6}$$

Since the above KB induces a partition of the concept $\mathsf{Student}$ between the concept $\mathsf{UGStudent}$ and the concept $\mathsf{GStudent}$, whenever two of these concepts are completely specified (e.g., they are DBox concepts) then also the third is necessarily completely specified.

It is possible to determine whether a concept or role is completely specified by a set of DBox predicates in a KB using only standard description logic inferences [14,8].

3.4 Constraints

As our proposal is quite different from the previous proposals for integrity constraints, we will provide examples covering the major use cases for integrity constraints and several variants of their variants.

The entire extension of completely specified concepts and roles is known by name. This makes it quite obvious that completely specified concepts and roles naturally enforce constraints concerning knowing the identity and type of role fillers. For example, if hasSpouse is a completely specified role, then any fillers of hasSpouse will be known by name. If this is the case, the axiom

$$\mathsf{MarriedPerson} \sqsubseteq {=}1\,\mathsf{hasSpouse} \tag{7}$$

will be true in a KB only if each married person has precisely one known spouse. So, the KB

$$\begin{aligned} \mathcal{T} &: \mathsf{MarriedPerson} \sqsubseteq {=}1\,\mathsf{hasSpouse.Person} \\ \mathcal{A} &: \mathsf{MarriedPerson(Joe)} \\ &\ \mathsf{MarriedPerson(Jack)} \\ \mathcal{DB} &: \end{aligned} \tag{8}$$

hasSpouse

Jack	Elizabeth
Jack	Liz

is unsatisfiable because Joe has no spouse (because he is distinct from Jack) and Jack has both Elizabeth and Liz as spouses (and they are distinct from each other). The situation would be completely different if hasSpouse was not a DBox role, in which case Joe would have been inferred to have some spouse, and Elizabeth and Liz would have been inferred to be the same.

Let's see now an example of an entailed constraint. Consider the KB

$$\begin{aligned} \mathcal{T} &: \mathsf{MarriedPerson} \sqsubseteq {\geqslant}1\,\mathsf{hasSpouse.Person} \\ \mathcal{DB} &: \end{aligned} \tag{9}$$

hasSpouse

Jack	Elizabeth

From the above, we can entail the statement

$$\mathsf{MarriedPerson} \sqsubseteq {=}1\,\mathsf{hasSpouse.Person}$$

which couldn't be derived if hasSpouse were not a completely specified role.

All entries in completely specified concepts and roles have to have a name, i.e., not be some unknown filler, eliminating one problem with minimal models approaches (KB 2.2_m). Nor is it possible for disjunctive information to be adequate, eliminating another problem with minimal models approaches. (KB 2.5_m). Here our approach has the desirable behavior of autoepistemic approaches, requiring known certain fillers. Similarly, there is no issue with whether unknown objects are considered by constraints, which causes problems for autoepistemic approaches (KB 2.4). Here our approach has the desirable behavior of autoepistemic approaches, with constraints effectively taking into account all possible interpretations, because there is only one.

In the previous example, it was not necessary that the spouse be known to be a person, because Person was not a completely specified concept. If however, all people are known, then Person would be a completely specified concept, and spouses of married people would need to be known as people. It may be the case, however, that not all people are known, only those that are spouses. In this situation, MarriedPerson would be a completely specified concept,

$$\begin{aligned}\mathcal{T}: &\ \mathsf{MarriedPerson} \sqsubseteq {=}1\,\mathsf{hasSpouse.Person}\\ &\ \mathsf{MarriedPerson} \sqsubseteq {=}1\,\mathsf{hasSpouse.MarriedPerson}\\ &\ \mathit{Sym}(\mathsf{hasSpouse})\end{aligned} \qquad (10)$$

$\mathcal{DB}$: hasSpouse

Jack	Elizabeth
Elizabeth	Jack

MarriedPerson

Jack
Elizabeth

Here any married person has to have precisely one spouse that is also a married person, as well as precisely one spouse that is a person. The extension of the concepts spouse and MarriedPerson provide precisely one known spouse that is known to be a married person as well as precisely one spouse overall for both Jack and Elizabeth, satisfying the constraint portions of these axioms. Because Jack and Elizabeth are each other's only spouse, they are both are inferred to belong to Person as well.

The situation where only a portion of a concept or role is completely specified and thus causes constraint-like behavior, is quite natural. For example, neither all people nor their SSN's may be known, but all taxpayers and their SSN's are. The following KB captures this situation:

$$\begin{aligned}\mathcal{T}: &\ \mathsf{Person} \sqsubseteq (\leqslant 1\,\mathsf{hasSSN}) \sqcap (\forall \mathsf{hasSSN.integer})\\ &\ \mathsf{TaxPayer} \sqsubseteq (\mathsf{Person}) \sqcap ({=}1\,\mathsf{hasSSNTP.integer})\\ &\ \mathsf{hasSSNTP} \sqsubseteq \mathsf{hasSSN}\\ \mathcal{A}: &\ \mathsf{Person(Jill)}\\ &\ \mathsf{Person(Susan)}\\ &\ \mathsf{hasSSN(Jill, 987654321)}\end{aligned} \qquad (11)$$

$\mathcal{DB}$: TaxPayer

Jack
...

hasSSNTP

Jack	123456789
...	...

Here it does not matter that Susan's SSN is not known, but Jack's must be, and so must that of all the other taxpayers.

Because there is no minimization involved in our approach, there is no problem with extra axioms modifying the satisfaction of constraints. As occurs in the minimal models approaches (KB 4 and variants). In

$$\begin{aligned}
\mathcal{T}:\ & \mathsf{MDPerson} \equiv \mathsf{Person} \sqcap \geqslant 2\, \mathsf{hasDependent.Child} \\
& \mathsf{FDPerson} \equiv \mathsf{Person} \sqcap \leqslant 2\, \mathsf{hasDependent} \\
\mathcal{A}:\ & \mathsf{MDPerson(Joe)} \\
& \mathsf{FDPerson(Jack)} \\
\mathcal{DB}:\ &
\end{aligned} \tag{12}$$

hasDependent	
Joe	Mary
Joe	Susan
Jack	Bill
Jack	John
Jack	Thomas

Child
Mary
Susan
Bill
John
Thomas

Mary and Susan are distinct, and no axioms can affect this situation, so Joe has two suitable dependents and the constraint axiom for MDPerson is satisfied on Joe regardless of anything else in the KB. Similarly, Bill, John, and Thomas are distinct so the constraint axiom for FDPerson is violated on Jack.

No disjunctive information can infect completely specified concepts or roles. In KB (3) about students, it is most likely that all of UGStudent, GStudent, major, and faculty are desired to be completely specified. If no other information is added, as in

$$\begin{aligned}
\mathcal{T}:\ & \mathsf{Student} \equiv \mathsf{UGStudent} \sqcup \mathsf{GStudent} \\
& \mathsf{UGStudent} \sqsubseteq \exists \mathsf{major} \\
& \mathsf{GStudent} \sqsubseteq \exists \mathsf{faculty} \\
\mathcal{A}:\ & \mathsf{Student(Mary)} \\
\mathcal{DB}:\ &
\end{aligned} \tag{13}$$

UGStudent

GStudent

major

faculty

then the KB is inconsistent, as Mary is neither an undergraduate nor a graduate, violating the first axiom.

The KB cannot be consistent without having Mary's situation as an undergraduate or graduate be provided, and then the required information about either her major or faculty, as in

$$\begin{aligned}
\mathcal{T}:\ & \mathsf{Student} \equiv \mathsf{UGStudent} \sqcup \mathsf{GStudent} \\
& \mathsf{UGStudent} \sqsubseteq \exists \mathsf{major} \\
& \mathsf{GStudent} \sqsubseteq \exists \mathsf{faculty} \\
\mathcal{A}:\ & \mathsf{Student(Mary)} \\
\mathcal{DB}:\ &
\end{aligned} \tag{14}$$

UGStudent
Mary

GStudent

major	
Mary	Psychology

faculty

Our approach can naturally handle disjunctive information that interacts with completely specified concepts and roles. For example, if KB (13) is modified so that major and faculty are completely specified (admittedly not a very normal setup) as follows:

$$\begin{aligned}
\mathcal{T} :\ & \mathsf{Student} \equiv \mathsf{UGStudent} \sqcup \mathsf{GStudent} \\
& \mathsf{UGStudent} \sqsubseteq \exists \mathsf{major} \\
& \mathsf{GStudent} \sqsubseteq \exists \mathsf{faculty} \qquad (15) \\
\mathcal{A} :\ & \mathsf{Student}(\mathsf{Mary}) \\
\mathcal{DB} :\ & \underline{\mathsf{major}} \qquad \underline{\mathsf{faculty}}
\end{aligned}$$

then the KB is still inconsistent. Mary does not have to be either known to be an undergraduate or known to be a graduate, but because of the disjunctive definition of Student she does have to be either an undergraduate, in which case she has no major, or a graduate, in which case she has no faculty. Both cases lead to an inconsistency.

4 RDF(S) and DBoxes

Our development of constraints using DBoxes and completely specified concepts and roles has used OWL (or $\mathcal{SROIQ}(\mathcal{D})$) as the ontology language. We used OWL for two reasons. First, previous work on constraints in ontology languages has concentrated on OWL or other expressive ontology languages, so using OWL here allows us to make better comparisons with previous work. Second, DBoxes can be rewritten as other OWL constructs, showing that DBoxes do add not any expressive power to OWL.

It is possible to use RDF or RDFS [3] as the ontology language for DBoxes. The basic idea is that for any URI D declared to be a DBox predicate in $\mathcal{S}(\mathcal{DB})$, the set of the URIs C_i stated *explicitly* in the graph as having D as their `rdf:type` – namely the set of all C_i appearing in triples of the form (C_i `rdf:type` D) – has to be considered as the *complete* set of instances of D. DBoxes extend the expressive power of RDF and RDFS because in DBoxes is it possible to infer that facts are false, e.g., any non-stated fact for a DBox concept or role. This addition of expressive power makes reasoning in RDF(S) plus DBoxes harder than reasoning in just RDF(S); as a matter of fact, we can prove the following theorem.

Theorem (Complexity of RDF(S) with DBoxes). *SPARQL query answering with basic graph patterns (BGPs) under the RDF simple entailment regime, and the RDFS entailment regime,* augmented with DBoxes *is coNP-hard for data complexity.*

The proof is based on a reduction to the 3-colorability problem, a reduction similar to the one employed in [7].

The use of DBoxes permits RDFS to represent the full meaning of database information that is imported into RDFS, adding an important aspect to RDFS. Even with the expressive weakness of RDFS, it is possible to make the kind of inferences that have been argued to be problematic, for example by using role domains or ranges to infer concept membership. The use of DBoxes turn role domain and range axioms into constraints, allowing the elimination of these inferences in cases where they might be problematic.

5 Conclusion

There is no perfect approach to integrity constraints in an ontology setting. One would like to have the situation in databases, where the behavior of integrity constraints is precisely a check against data, while still retaining the open nature of ontology languages. This is not possible because the open nature of ontology languages means that axioms make inferences and do not simply check the data. One would also like to be able to check integrity constraints quickly, but this is also not possible. In an open setting checking an integrity constraint is at least as costly as determining the consistency of the KB.

Previous approaches to integrity constraints involving autoepistemic constructs use the autoepistemic constructs to determine what is known (in effect, closing off the current knowledge) and utilize this closed version to ensure that the integrity constraints do not add new information. However, autoepistemic approaches make the constraints operate on too few individuals, e.g., only on known instances of a concept, which limits their ability to truly check that the constraints hold.

Approaches involving minimal models have different problems. Instead of having the constraints active on too few individuals, the constraints are too easy to satisfy. Without the use of a special predicate, existential, or unknown, fillers are acceptable in constraints. Even with the solution to the previous problem, disjunctive information is adequate to satisfy a constraint. Further, the minimization needed in the approach is sensitive to the presence of extra concepts in the KB, and axioms that should be irrelevant can change whether a constraint is satisfied or not.

Approaches that involve both minimization (for example of just equality) and modal notions fall prey to versions of the two different approaches. Modal evaluation of constraints means that the constraints often are active on too few individuals. Even minimization of just equality is sensitive to irrelevant axioms.

Our approach to constraints is to completely specify certain concepts and roles, making them into the analogue of database tables. On these concepts and roles, axioms act just like integrity constraints. Although this approach may appear to be without computation cost, the extensive use of nominals in the translation of DBoxes to regular ontology languages can easily stress current ontology reasoners. As well, because DBoxes are closed, adding new information to a DBox concept or role is a modification of the KB, not just an addition.

We hope that future work on ontology reasoners will provide optimizations for both extensive use of nominals and modifications to the KB. Both of these situations occur commonly, and are not specific to DBoxes.

On the other hand, querying DBox concepts and roles is easy, as it is just database querying. Further, answers returned by such queries are complete (as opposed to the situation with other concepts and roles, where the information returned might not be complete) and can be used in applications just like answers to database queries can.

Even with the computational issue, we believe that the DBox approach is the correct approach to providing integrity constraints for ontology languages. The DBox approach provides precisely the same effect for integrity constraints as is the case in databases, which provide the model for integrity constraints. It thus avoids the problems with the other approaches, and thus appears to us to be preferable.

References

1. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F. (eds.): The Description Logic Handbook: Theory, implementation, and applications, 2nd edn. Cambridge University Press (2010)
2. Beth, E.W.: On Padoa's methods in the theory of definitions. Indagotiones Mathematicae 15, 330–339 (1953)
3. Brinkley, D., Guha, R.V.: RDF vocabulary description langauge 1.0; RDF schema. W3C Recommendation (February 2004), http://www.w3.org/TR/rdf-schema/
4. de Bruijn, J., Polleres, A., Lara, R., Fensel, D.: OWL Flight. Deliverable D20.3v0.1, WSML (August 2004), http://www.wsmo.org/TR/d20/d20.3/v0.1
5. de Bruijn, J., Polleres, A., Lara, R., Fensel, D.: OWL DL vs. OWL Flight: Conceptual modeling and reasoning on the semantic web. In: Proceedings of the Fourteenth International World Wide Web Conference (WWW 2005), pp. 623–632 (May 2005)
6. Donini, F.M., Nardi, D., Rosati, R.: Description logics of minimal knowledge and negation as failure. ACM Transactions on Computational Logic 3(2), 177–225 (2002)
7. Franconi, E., Ibáñez-García, Y.A., Seylan, I.: Query answering with DBoxes is hard. Electronic Notes on Theoretical Computer Science 278, 71–84 (2011)
8. Franconi, E., Kerhet, V., Ngo, N.: Exact Query Reformulation with First-Order Ontologies and Databases. In: del Cerro, L.F., Herzig, A., Mengin, J. (eds.) JELIA 2012. LNCS, vol. 7519, pp. 202–214. Springer, Heidelberg (2012)
9. Horrocks, I., Kutz, O., Sattler, U.: The even more irresistible SROIQ. In: Doherty, P., Mylopoulos, J., Welty, C.A. (eds.) Proceedings, Tenth International Conference on Principles of Knowledge Representation and Reasoning, pp. 57–67. AAAI Press (June 2006)
10. Motik, B., Horrocks, I., Sattler, U.: Bridging the gap between OWL and relational databases. Journal of Web Semantics 7(2), 74–119 (2009)
11. Motik, B., Patel-Schneider, P.F., Parsia, B.: OWL 2 web ontology language: Structural specification and functional-style syntax. W3C recommendation (October 2009), http://www.w3.org/TR/owl2-syntax
12. Motik, B., Rosati, R.: Reconciling description logics and rules. Journal of the ACM 57(5), 1–62 (2010)
13. Sengupta, K., Krisnadhi, A.A., Hitzler, P.: Local Closed World Semantics: Grounded Circumscription for OWL. 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. 617–632. Springer, Heidelberg (2011)
14. Seylan, I., Franconi, E., de Bruijn, J.: Effective query rewriting with ontologies over DBoxes. In: Boutilier, C. (ed.) Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI 2009), pp. 923–929 (July 2009)
15. Tao, J., Sirin, E., Bao, J., McGuinness, D.L.: Integrity constraints in OWL. In: Proceedings of the Twenty-Fourth National Conference on Artificial Intelligence, Atlanta, Georgia. American Association for Artificial Intelligence (July 2010)

A Machine Learning Approach for Instance Matching Based on Similarity Metrics

Shu Rong[1], Xing Niu[1], Evan Wei Xiang[2], Haofen Wang[1], Qiang Yang[2], and Yong Yu[1]

[1] APEX Data & Knowledge Management Lab, Shanghai Jiao Tong University
{rongshu,xingniu,whfcarter,yyu}@apex.sjtu.edu.cn
[2] Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Hong Kong
{wxiang,qyang}@cse.ust.hk

Abstract. The Linking Open Data (LOD) project is an ongoing effort to construct a global data space, i.e. the Web of Data. One important part of this project is to establish `owl:sameAs` links among structured data sources. Such links indicate equivalent instances that refer to the same real-world object. The problem of discovering `owl:sameAs` links between pairwise data sources is called *instance matching*. Most of the existing approaches addressing this problem rely on the quality of prior schema matching, which is not always good enough in the LOD scenario. In this paper, we propose a schema-independent instance-pair similarity metric based on several general descriptive features. We transform the instance matching problem to the binary classification problem and solve it by machine learning algorithms. Furthermore, we employ some transfer learning methods to utilize the existing `owl:sameAs` links in LOD to reduce the demand for labeled data. We carry out experiments on some datasets of OAEI2010. The results show that our method performs well on real-world LOD data and outperforms the participants of OAEI2010.

Keywords: Linking Open Data, instance matching, similarity matric, machine learning, transfer learning.

1 Introduction

Linked Data[4] is a way to construct a global data space, the Web of Data, by interconnecting many structured data sources within the Linking Open Data[1] (LOD) project. These data sources are published under the Resource Description Framework[2](RDF). Each of them may contain millions of RDF triples.

The main idea of Linked Data is to construct typed links between different data sources. Such links describe the relationships between things so that users can browse data among sources by navigating along the links and agents can provide expressive query capabilities over the data on the web just like a local

[1] http://linkeddata.org/
[2] http://www.w3.org/RDF/

database. An important link is `owl:sameAs`[3] which indicates that the two instances it links refer to the same real-world object. Different data sources may have different emphases in describing things. Various descriptions can be aggregated according to the `owl:sameAs` links.

Currently, there are more than 300 data sources in LOD while there were only twelve of them in 2007 when the project started. As more and more data sources emerge, there is an urgent demand to provide `owl:sameAs` links from the new data sources to the existing ones. At the same time, the existing links in LOD are not as extensive as one would hope[17]. Instance matching is a practical idea for constructing such links. In general, the existing approaches for matching instances in LOD can be divided into two types. One is based on rules and the other is based on the similarity metric of instance pairs. Most of these methods do not always work well in the LOD scenario since they depend on the result of property matchings. Property matching links the properties from different data sources which have similar semantics such as `foaf:name` and `dc:title`. Property matching is not trivial since the data sources usually design their own ontologies to describe things. Furthermore, we noticed that although some properties in heterogeneous ontologies can not match, they have some connotative relationships. Their values may be worth considering for instance matching. For example, Freebase[4] says that the `fb:profession` of Steve Jobs is "Chief Executive Officer"and his `fb:place_of_death` is "Palo Alto", whereas DBpedia[5] says his `dbp:residence` is "Palo Alto California"and the information about "Chief Executive Officer"is in the text of the `dbp:abstract`. Such information will be ignored in property matching based methods, although it could be significant for human beings to judge whether the two Jobses match. We are inspired to explore the "common-sense "used for matching instances. The goal of this paper is to develop an automated instance matching method that is "common"and provides high accuracy. Such method should be independent of property matching to achieve "commonality".

In this paper, we employ machine learning models for instance matching based on some similarity metrics of instances. The matching instance pairs may have some common features in the similarity metrics of each pair. For example, two matching instances may share some significant words such as "Palo Alto"and "Chief Executive Officer"which we mentioned above, while the non-matching ones may not. Sharing some significant words is a common feature of the matching instance pairs here. We design a similarity vector independent of property matching to represent such features. Based on this vector, we train a learning model to classify the instance pairs as matching or non-matching. To minimize the demand for training data and promote the performance, we try to use existing instance matching information in LOD for help. A transfer learning method is applied to implement this idea.

[3] http://www.w3.org/TR/2004/REC-owl-semantics-20040210/#owl_sameAs/
[4] http://www.freebase.com/
[5] http://www.dbpedia.org/

We tried our approach on real LOD data sources which were chosen for IM@OAEI2010[6]. Our performance is better than the participating teams'. Another comparative experiment shows that the existing matching information can really help matching instances from the new data source pairs.

The following technical contributions are made:

– We utilize the values of non-matching properties for instance matching. Such values can be useful but are usually ignored by the existing instance matching approaches.
– We propose a novel approach for instance matching which is independent of property matching with high accuracy.
– We use existing `owl:sameAs` links to help match instances. Due to the heterogeneous ontologies constructed by various data sources, such information is hardly utilized in the existing instance matching approaches.

The remainder of this paper is structured as follows. Section 2 gives some definitions about instance matching and an overview of our proposed approach. Section 3 describes the feature extraction process. The selection of machine learning models is discussed in Section 4. Experimental results on the LOD datasets from LOD are reported in Section 5. Some related work is discussed in Section 6. Finally, Section 7 concludes this paper and discusses future work.

2 Definition and Framework

2.1 Problem Definition

An instance consists of some property-value descriptions about a real-world object. A pair of distinct instances a and b match if they refer to the same object, denoted by $(a, b) \in \mathcal{R}$. In LOD, `owl:sameAs` links are established between matching instances. When establishing such links, we usually consider a pair of data sources each time.

Definition 1 (Instance Matching). *Given two data sources A and B as input, the goal of instance matching is to compute the set $\mathcal{M} = \{(a, b)|(a, b) \in A \times B, (a, b) \in \mathcal{R}\}$.*

According to the definition of instance matching, the problem of finding matching instance pairs can be formalized as a binary classification problem.

Definition 2 (Instance Matching as Binary Classification). *Given two data sources A and B, the goal of instance matching is to find a classifier $\mathcal{C} : (a, b) \to \{-1, 1\}$ for $(a, b) \in A \times B$ such that $\mathcal{C}$ maps the non-matching instance pairs to class -1 and the matching ones to the class 1.*

The binary classification problem can be solved by traditional machine learning algorithms, which require multidimensional features as the input. In the problem of instance matching, we extract a feature vector from each instance pair (a, b).

[6] `http://oaei.ontologymatching.org/2010/im/index.html`

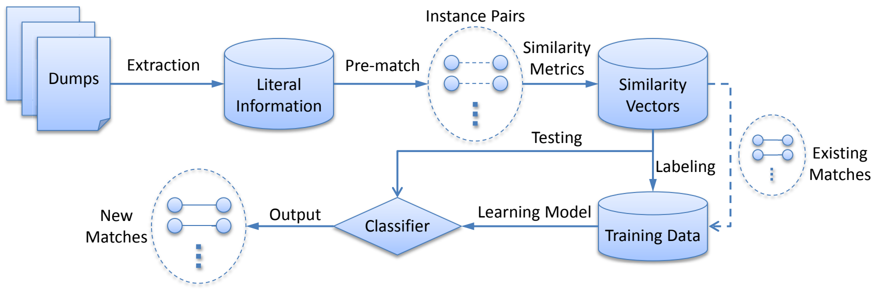


Fig. 1. Overview of the framework

Definition 3 (Similarity Vector of Instance Pairs). *The n-dimensional feature vector v of an instance pair (a, b) consists of n various similarities of instance a and b. Dimension $v_i = d_i(a, b)$, where d_i is the ith similarity metric function for (a, b).*

The feature vector of an instance pair indicates the similarities of the two instances, which are computed by several metric functions. Some existing instance matching (record linkage[11]) approaches also extract such a feature vector for classification, while each d_i computes the similarity of two values, one from each instance, that belong to a pair of matching properties (fields). Unlike these approaches, our similarity metric functions are based on the property-independent literal information extracted from each instance. The literal information $l = \{l_1, l_2, \ldots, l_n\}$ is similar to a virtual document generated from an instance. For an instance pair (a, b), a similarity metric function d_i maps the extracted literal information pair (l_i^a, l_i^b) to a real number in the range of $[0, 1]$.

2.2 Framework

The framework of our approach is shown in Figure 1. We extract literal information for each instance from the property-value descriptions. To get sufficient literal information for the similarity metrics, we conduct the following preprocessing for each property-value pair:

– A real-world object may be represented by an instance or a piece of text. For consistency, if the value is a URI which represents another instance, we will replace it by the label of that instance. Most of the instances have a label value which usually belongs to the property `rdfs:label` or some other common properties[10]. If the label of an instance is unavailable, we can replace it by its text description.
– We will also replace each property by its label. If the label is unavailable, we will use the last token (normalized) of the URI instead, e.g., "place of death "for `fb:place_of_death`.

For data sources A and B which contain $|A|$ and $|B|$ instances respectively, there are $|A| \times |B|$ possible instance pairs. This is unacceptable since both $|A|$ and $|B|$ can be one million or even larger. We use a simple pre-match method to sift the possible matching pairs. An inverted index is built for instances of some key words in their descriptions. The instances sharing the same keys in the index are considered to be candidate matching instances. Our preliminary experiments show that this sifting method can greatly reduce the number of pairs to test for a match, without losing much recall (the recall is over 0.9).

After preprocessing and literal information extraction, we compute the similarity vectors for the candidates. To train a binary classifier based on the similarity vectors, we need to label some of them into class -1 (non-matching) or class 1 (matching). Thus the problem of instance matching can be solved by classifying the unlabeled instance pairs. The existing matching instance pairs in LOD can be considered as labeled data which may help to train the classifier. We employ a transfer learning algorithm to improve the effect of the helping.

3 Feature Extraction

We extract some property-independent information from each instance and then compute the similarity vectors for instance pairs based on this information. Since the performance of a machine learning algorithm depends a lot on the quality of feature extraction, the work in this section is vital for the next step.

3.1 Literal Information Extraction

We extract several sets of literal information from each instance. The first is the text information set l_{label}, that is the label of an instance. The label is the human-readable name for an instance, such that it can help people to identify the real-world object. So labels are discriminative for instance matching. Next, we extract the remaining text information from the instance. These sets are divided into two parts. One is $l_{property}$ which consists of the text information from properties. The other is the text information from the values. The number of words in a value has a certain meaning. If the value only contains one word, this word can be a specific symbol such as the ISBN for a book. If the number of words is small, these words are likely to be the name of something, e.g. "Palo Alto". If there are a lot of words in the value, they may be a text description. These three kinds of values may play different roles in the problem of instance matching. So we extract them as l_{single}, l_{short} and l_{long} respectively.

Besides the large amount of text information, there are also other types of literal information in the instance descriptions. The common ones we used are dates, numbers and links. In contrast to the text information, these types of literal information are more useful for instance matching. If two instances share some dates, numbers or links, they are likely to match. So we additionally extract them as l_{date}, l_{number} and l_{link}. Note that:

Table 1. Overall Statistics on Extraction Results

Dimension Num	Metric Function	Combination of Literal Information
1	IdfSim	l_{single}
2	TopIdfSim	l_{single}
3	IdfSim	$l_{single} \cup l_{short} \cup l_{label}$
4	TopIdfSim	$l_{single} \cup l_{short} \cup l_{label}$
5	CosSim	$l_{single} \cup l_{short} \cup l_{label} \cup l_{property} \cup l_{long}$
6	IdfSim	$l_{single} \cup l_{short} \cup l_{label} \cup l_{property} \cup l_{long}$
7	TopIdfSim	$l_{single} \cup l_{short} \cup l_{label} \cup l_{property} \cup l_{long}$
8	EditSim	l_{label}
9	CountSim	l_{label}
10	CountSim	l_{date}
11	CountSim	l_{number}
12	CountSim	l_{link}

– There are many forms of dates. For convenience, we only extract the year part of each date and the other parts are treated as text.
– Some dates and numbers may be included in the texts. We use meticulous string processing to find them.

3.2 Similarity Metrics

Different similarity metric functions are used for different types of literal information. As shown in Table 1, a 12-dimensional similarity vector is generated for each instance pair.

For the text information, we use three functions, CosSim, IdfSim and TopIdfSim. CosSim is a common similarity metric for texts. It computes the TF · IDF[7] weights for the words from two word sets and then computes their *cosine similarity*. Furthermore, in the particular problem of instance matching, the IDF weights are more important. Some common words or common words for the domain may appear frequently in the descriptions of many instances. These words with high TF weights but low IDF weights do not much help match instances. While if a word only appears once in each data source, the two instances that contain it are likely to match. According to this idea, IdfSim and TopIdfSim are designed based on the IDF weights of words. IdfSim is similar to CosSim which just removes the TF weights. For word sets T_1 and T_2, TopIdfSim computes the similarity of W_1 and W_2, where W_i is a subset of T_i which consists of the words with highest IDF weights in T_i. It is computed by:

$$\text{TopIdfSim}(T_1, T_2) = \frac{\sum_{w \in W_1 \cap T_2} \text{IDF}(w) + \sum_{w \in W_2 \cap T_1} \text{IDF}(w)}{\sum_{w \in W_1} \text{IDF}(w) + \sum_{w \in W_2} \text{IDF}(w)} \qquad (1)$$

These three similarity metric functions act on the combinations of the extracted word sets of text information. The combining strategy is based on the relaxed inclusion relation of these word sets from different instances, that is l_{single} may be included in l_{short} or l_{label}, and l_{long} main contains all the other word sets.

Among the sets of text information, l_{label} is different from the others in two ways as follows:

1. We only extract one label for each instance; so it can be treated as a set of words or a string.
2. Since each word in a label can be significant for recognizing the entity, to match two instances, the matching words of their labels are more important than the non-matching ones.

So we design another two similarity metrics for l_{label}, which are EditSim and CountSim.

$$\text{EditSim}(l^a_{label}, l^b_{label}) = 1 - \frac{\text{EditDistance}(S_a, S_b)}{\text{Max}(|S_a|, |S_b|)} \tag{2}$$

Where S_a stands for the string form of l^a_{label} and EditDistance(S_a, S_b) is a classical string-distance measure, which represents the minimum number of editing operations needed to make S_a and S_b the same. Each operation can be deleting a character, inserting a character or changing a character in either S_a or S_b.

$$\text{CountSim}(l^a_{label}, l^b_{label}) = \frac{1 - 2^{-|\{w|w \in l^a_{label} \cap l^b_{label}\}|}}{1 - 2^{-\lceil (|l^a_{label}| + |l^b_{label}|)/2 \rceil}} \tag{3}$$

The literal information sets of dates, numbers and links also have the second characteristic of the labels. So we use CountSim on them to generate similarities. Note that two numbers are considered to be the same one if their difference is lower than a threshold.

4 Learning Algorithms

After extracting the feature vectors, we can train the classifier for instance matching. There are many machine learning algorithms for the binary classification problem. We need to carefully choose the appropriate ones according to the characteristic of the input data.

4.1 Basic Learning Model

In our problem, the input data may contain millions instance pairs. So some methods with high time cost such as *Neural Networks* and *SVM* with a complex kernel are eliminated. After observing the feature space via a preliminary experiment, we found that the data of the two classes are not linearly separable. A typical example is shown in Figure 2. x and y are two dimensions of the similarity vector. The positive and negative regions represent the two classes. Figure 2 indicates that for a similarity vector, if x is greater than a threshold, it belongs to the class of matching instance pairs when y is large, but if x is less than the threshold, it belongs to the matching class when y is small. From the

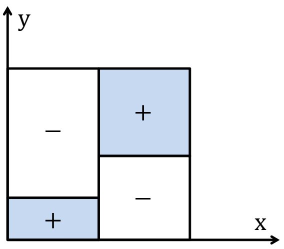


Fig. 2. A typical example of the feature space

perspective of each single dimension, this situation does not meet our intuition. The value of each dimension describes the similarity of two instances based on a certain metric. The higher the value is, the more likely they will match. But from the perspective of the correlation between the dimensions, such a situation is reasonable. We will give an example to explain it.

On one hand, given two instance pairs (p_1, p_2) and (q_1, q_2), their similarity vectors are u and v. We assume that $u_6 = v_6 = 0.3$, $u_5 = 0.1$ and $v_5 = 0.3$ where $u_i(v_i)$ stands for the ith dimension of the similarity vector $u(v)$. Probably, instance p_1 consists of long texts and instance p_2 consists of short texts or single words, so that $u_5 = 0.1$. While q_1 and q_2 both consist of long texts which share some common (for the domain) words, so $v_5 = 0.3$. Furthermore, p_1 and p_2 may share some important words such that $u_6 = 0.3$. While the value 0.3 of v_6 may be obtained by the common words. According to the inference above, (p_1, p_2) is likely to match while (q_1, q_2) is not. On the other hand, an instance pair with large values of both dimensions 5 and 6 is likely to match.

Since the feature space is so complex, some linear classifiers are inapplicable here, e.g. *linear regression* and *SVM*. Actually, the dimensions of our similarity vector have some implicit associations, especially the ones generated from the same combination of text sets. So the model we needed should properly handle such associations for classification.

Consider the *decision tree* model: A decision tree is a binary tree. Each non-leaf node t in the tree has a dimension-number k_t and a threshold σ_t and each leaf node has a label of one of the two classes. When a testing vector is sent to a non-leaf node t, it will be sequentially sent to a child node according to whether the value of k_tth dimension of the vector is greater than σ_t. So a testing vector will be sent to the root of the tree and finally get to a leaf node. The label of the leaf node will be the class the vector belongs to. The vectors which arrive at node t have passed the ancestors of t, i.e. A_t. In this way, the associations between dimension of k_t and the dimensions of $\{k_a | a \in A_t\}$ are taken into consideration for classification.

4.2 AdaBoost with Random Forest

AdaBoost[12] is the predecessor of the transfer learning algorithm we will use. It combines several weak classifiers to get a powerful classifier via an iterative process.

In each round of iteration, it employs a basic learning model to train a weak classifier with the current weighted examples. Then it increases the weights of the examples which are incorrectly classified by the weak classifier, so that these "difficult" ones will be classified better in the following rounds. In this way, *AdaBoost* may improve the performance of the basic learning model.

We found that some classical decision tree algorithms do not show good performance working with *AdaBoost*. In the training phase, the decision tree will continually grow to satisfy the training data. Since *AdaBoost* combines the examples from source and target domain for training, the trained tree-classifier will achieve good performance on both source and target domain examples. This leads to a situation where few incorrectly classified examples can be found during the iterations and the weights are hardly changed.

To solve this problem, we choose a specific decision tree model, *Random Forest*. A random forest contains several decision trees. Each of them is trained with a newly constructed training set, which is chosen by randomly picking some examples with replacement from the whole training data. The classification result of the random forest is a voting result of these trees. Due to the chosen strategy of the training set, most trees will only be good at classifying the examples in the dominating distribution. The remaining examples will be incorrectly classified so that they will be reweighted. Our preliminary experiments show that as a basic learning model of *AdaBoost*, *Random Tree* is superior to the other decision tree models, e.g. *J48Tree*[20].

4.3 Transfer Learning

To train an efficient classifier, a number of training examples are needed and should be labeled manually. To reduce the manual work, we want to utilize the existing matching instance pairs in LOD for help. But most machine learning methods work well only under the assumption that the training and testing data are drawn from the same feature space and the same distribution[24]. Training data which is generated from the existing matching instance pairs does not meet this requirement. *Transfer learning*[24] can utilize the data from different distributed domains (*source domain*) to help the target task, thus reducing the need for training data in the *target domain* (the domain of testing data).

There are two main kinds of transfer learning methods: the *instance-transfer* and the *feature-representation-transfer*. The former assumes that the feature spaces of the source and target domain are the same while the distributions of the data from the two domains are different. Such methods try to find examples from the source domain which can be reused to help the target domain task. The latter assumes that the feature spaces of the source and target domain are different. Such methods focus on finding the "good" common features which reduce the difference between the source and target domains and the error of classification.

The property matching independent similarity vectors we generated naturally compose a common feature space for all the data source pairs. So we employ *TrAdaBoost*[8] for help, which is a classical algorithm for *instance-transfer* learning. *TrAdaBoost* is an extension of the *AdaBoost*[12] algorithm. It assumes that the feature spaces of source and target domain are the same while the distributions of the data from the two domains are different. Due to the different distributions, some of the source domain data may be helpful in training the classifier for the target domain but some of it may not be and could even be harmful. *TrAdaBoost* iteratively adjusts the weighting of the source domain data to reduce the effect of the "bad"source data. In each round of iteration, *TrAdaBoost* trains the basic classifier on the weighted source and target data. The source domain examples which are classified incorrectly by the classifier are considered to be the bad source data, so their weights are reduced. Meanwhile, *TrAdaBoost* uses the same strategy as *AdaBoost* that is to increase the weights of the incorrectly classified examples in the target domain.

Random Forest is a suitable basic learning model for *TrAdaBoost*, as well as for *AdaBoost*. The interface of *TrAdaBoost* is generic. We can directly apply it on the problem of instance matching by treating the instance pairs from a pair of data sources, which are to be matched, as the target domain, and the existing matching information between another pair of data sources as the source domain. But not all the source domains can help to train the classifier on the target domain via *TrAdaBoost*. A source domains can be harmful if its distribution is quite different from that of the target domain.

The problem of how to automatically choose a helpful source domain has not been theoretically solved yet[9]. Intuitively, the more similar the distributions of the two domains are, the more likely the source domain can help. In the problem of instance matching, the distribution of a domain is decided by the heterogeneity between the ways of describing objects which are used by the data sources (We call it describing heterogeneity for short). So when we want to match the instances of a data source pair (A, B), we should use another pair (C, D) for help, such that the describing heterogeneities of (A, B) and (C, D) are similar.

5 Experiments

First, we will show the experimental results of our proposed approach without transfer learning. We use the dataset provided by the data interlinking track of IM@OAEI2010 and compare our approach with the participants'. We chose this dataset because many others are not from LOD. Then we will give some comparative experiments to show whether a source domain we chose from LOD is helpful to instance matching via transfer learning.

5.1 Without Transfer Learning

The goal of the data interlinking track of IM@OAEI2010 is to find all the `owl:sameAs` links from four data sources to the ones in LOD. These four data sources are also in LOD; they are:

- Sider[7], about some drugs and their effects.
- DrugBank[8], about drugs and their chemical, pharmaceutical and pharmacological information.
- DiseaSome[9], about disorders and genes.
- DailyMed[10], about marketed drugs and chemical structure, mechanism of action, indication, usage, contraindications and adverse reactions for the drugs.

These data sources are already linked to LOD and the existing links will be treated as the standard answers. The well-known *Recall*, *Precision* and *F-Measure* are used for evaluation.

Two teams, ObjectCoref and RiMOM took part in this track. Their reports of results can be found in [32] and [16]. ObjectCoref[17] uses a self-learning framework to iteratively extend a kernel of matching instances. In each round of iteration, the most discriminative property-value pair is learned from the kernel for the further matching. RiMOM[19] is a multi-strategy ontology matching framework. It combines three strategies when facing an instance matching problem.

1. Edit distance between labels of two instances.
2. Cosine of the $TF \cdot IDF$ vectors for the text description of the instances to match.
3. Cosine of the $TF \cdot IDF$ vectors for the text description of the instances related to the ones to match.

In Table 2, we give the results of matching instances for each data source pair. RiMOM also gave their results on some other data source pairs which are not shown here, since we can not find the dumps of those data sources. For each testing data source pair, we randomly labeled 5% of the similarity vectors as training data (no more than 2000 for these datasets). Obviously, our proposed approach works better than ObjectCoref and RiMOM on these datasets.

For ObjectCoref, the process of learning discriminative property-value pairs depends on the lax matches of properties from different data sources. From the report of ObjectCoref[16], we can see that the matching properties found for these datasets are mainly about names and aliases. By analyzing the data, we find that Sider, DrugBank and DayliMed contain a lot of aliases for each instance, while DiseaSome does not. Furthermore, some non-matching instances have similar aliases. So ObjectCoref got high recall and low precision on Sider-DrugBank and Sider-DailyMed, but low recall and high precision on Sider-DiseaSome. In general, ObjectCoref did not get good performance since the properties of names and aliases do not match well. In contrast, RiMOM is a property matching independent approach. But the similarity metric that combines the three strategies is not accurate enough for instance matching.

[7] http://sideeffects.embl.de/
[8] http://www.drugbank.ca/
[9] http://http://diseasome.kobic.re.kr/
[10] http://dailymed.nlm.nih.gov/

Table 2. Compare with the Participants of IM@OAEI2010

Data Set	Approach	Recall	Precision	F-Measure
Sider-DrugBank	ObjectCoref	0.996	0.302	0.464
	RiMOM	0.342	0.961	0.504
	AdaBoost	0.859	0.952	**0.903**
Sider-DiseaSome	ObjectCoref	0.668	0.837	0.743
	RiMOM	0.315	0.837	0.458
	AdaBoost	0.726	0.875	**0.794**
Sider-DailyMed	ObjectCoref	0.999	0.548	0.708
	RiMOM	0.567	0.706	0.629
	AdaBoost	0.672	0.805	**0.733**
Sider-DBpedia	RiMOM	0.482	0.717	0.576
	AdaBoost	0.643	0.639	**0.641**
DailyMed-DBpedia	RiMOM	0.246	0.293	0.267
	AdaBoost	0.373	0.377	**0.375**

Table 3. Transfer GeoNames-DBpedia to LinkedGeoData-DBpedia

Training Examples	AdaBoost	AdaBoost(Source)	TrAdaboost
900	0.284	0.372	**0.378**
1500	0.383	0.396	**0.432**
3000	0.444	0.416	**0.458**
6000	**0.524**	0.450	0.516
15000	**0.544**	0.491	0.536

We noticed that our proposed approach has an enormous advantage on the data set Sider-DrugBank. The probable reason is that we can make use of the names and aliases for instance matching. Our approach eliminates the ill effects of the duplicate names by giving them low IDF weights.

5.2 With Transfer Learning

We choose GeoNames[11], LinkedGeoData[12] and DBpedia as the datasets for the experiments on transfer learning. GeoNames and LinkedGeoData are data sources in LOD. Both of them are about geographic information and have `owl:sameAs` links to DBpedia. GeoNames and LinkedGeoData have similar behaviors in describing real-world objects. So the the describing heterogeneities of GeoNames-DBpedia and LinkedGeoData-DBpedia are similar. We try to use the

[11] http://www.geonames.org/

[12] http://linkedgeodata.org/

information on the existing matching instances between GeoNames and DBpedia to help matching LinkedGeoData and DBpedia.

The result is shown in Table [3]. **AdaBoost** denotes the *AdaBoost* model applied only on the training data from the target domain (LinkedGeoData-DBpedia). **AdaBoost(Source)** and **TrAdaBoost** respectively denote the *AdaBoost* and *TrAdaBoost* model applied on the training data from both domains. **Training Examples** denotes the number of training instance pairs we labeled in the target domain. 900 examples are about 0.01% of all the pre-matching instance pairs between LinkedGeoData and DBpedia. We can see that the source domain we chose is really helpful via *TrAdaBoost*. But directly using the source domain for training can be harmful. Furthermore, the less training data there is in the target domain, the more the source domain can help. If there is efficient training data in the target domain, the source domain is entirely useless. These experimental results match our intuition about transfer learning.

6 Related Work

Although the problem of instance matching has emerged along with the development of LOD in recent years, a similar problem, *Record linkage*, was examined much earlier. Thus a lot of relevant approaches have been proposed.

6.1 Record Linkage

Record linkage, also known as *duplicate detection* or *object identification*, is a classical problem in the area of databases. The goal of record linkage is to determine the pairs of records that are associated with the same entity across various data sets. It is similar to the instance matching problem. For more than five decades, the traditional database community has discussed this problem a lot. Some surveys can be found in [33], [34] and [11].

The early approaches focus on similarity metrics for single field (column) matching. Many of them are even widely used today, such as edit distance, Q-gram distance, etc. The similarity vector in our work is based on $\mathrm{TF} \cdot \mathrm{IDF}$[7] and its variants.

The approaches for multi-field matching are mainly based on probabilistic models, developed by the machine learning community. The learning methods were applied on record linkage by extracting the feature vector of similarities from the comparable fields. Such approaches classify the record pairs as matching or not, employing CART[6], SVM[3] and so on. Semi-supervised, active and unsupervised learning algorithms have been proposed to reduce the demand for training data. Unlike the properties of linked data sources, the number of fields in a record linkage problem is usually quite small. Thus the comparable fields can be easily found manually. So the data base community doesn't pay attention to the problem known as *Structural heterogeneity*. A simple schema-independent method has also been proposed which treats the whole record as one large field. But experiments in [3] show that SVM based on the similarity metric of multiple comparable fields usually outperforms it. It's easy to see that such an elaborate similarity metric benefits record linkage when the fields can well match. Some distance-based methods which have also been proposed for for multi-field matching do not employ machine learning algorithms[13][1], but the distance measures are also based on schema matching.

Some other work focuses on postprocessing. The main idea is that the matching relations between records or fields should be consistent[2][25]. Such methods based on graph model can play an important role in instance matching problems as the linked data settings are more structured. They are complementary to our proposed work which uses only the literal information.

6.2 Instance Matching

Some of the existing links in LOD are discovered manually with some tools. One well-known instance matching tool is the Silk Link Discovery Framework[31] which allows setting link specifications for given data sources. Domain related knowledge is required to design such specifications.

The automatic instance matching approaches are often domain specific or property matching dependent. The approach proposed in [29] matches the FOAF instances using SVM. Since all of the instances are described with FOAF, the features for classification are easily determined according to the limited number of properties. More domain-specific approaches can be found in [27] and [28]. Among the domain-independent approaches, [14] matches instances according to their inverse functional properties (IFP). Such properties are not sufficient in LOD, so [15] tries to find more IFPs with a statistical method. ObjectCoref[17] employs a self-learning framework to iteratively find the discriminative property-value pairs for instance matching, which are lax IFPs. RAVEN[22] applies active learning techniques for instance matching. Both ObjectCoref and RAVEN match the properties from different data sources by measuring value similarities. Similar ideas are proposed in the domain of *schema matching*[26].

Finally, some papers focus on improving the efficiency of instance matching. [21] limits the number of candidate instance pairs to match based on the triangle equation. [30], [23] and [18] generate candidates by indexing some key words of instances. This kind of method can be applied to optimize ours.

7 Conclusion and Feature Work

In this paper, we presented a property matching independent approach for instance matching. We transformed the problem of instance matching to a classification problem, by designing a novel feature vector of high-level similarity metrics. Suitable learning models were selected according to the feature space. Our experimental results on the datasets of IM@OAEI2010 shown that such a feature vector is reasonable for instance matching, and our approach performed much better than the contest participants. Furthermore, we tried to utilize the information of existing matches in LOD to help match new data sources via a transfer learning algorithm. The experiments also show that such information is really helpful.

In future work, we will try to explore the following issues:

– The information on property matching and the relationships between instances can be taken into consideration in the similarity metrics. It may enrich the features for instance matching.
– *Random Forest* and *Adaboost* are similar in the result of classification[5]. Cooperation of *Random Forest* and *TrAdaboost* can be explored.
– The number of dimensions of the current similarity feature space is too low for machine learning with so many matching instances in LOD. More property matching independent similarity metrics need to be designed to make the best use of the information in the existing matches.
– We hope to find a powerful way to automatically choose a helpful source domain for the problem of instance matching.

References

1. Ananthakrishna, R., Chaudhuri, S., Ganti, V.: Eliminating fuzzy duplicates in data warehouses. In: Proceedings of the 28th International Conference on Very Large Data Bases, pp. 586–597. VLDB Endowment (2002)
2. Bansal, N., Blum, A., Chawla, S.: Correlation clustering. Machine Learning 56(1), 89–113 (2004)
3. Bilenko, M., Mooney, R., Cohen, W., Ravikumar, P., Fienberg, S.: Adaptive name matching in information integration. IEEE Intelligent Systems 18(5), 16–23 (2003)
4. Bizer, C., Heath, T., Berners-Lee, T.: Linked data-the story so far. International Journal on Semantic Web and Information Systems (IJSWIS) 5(3), 1–22 (2009)
5. Breiman, L.: Random forests. Machine Learning 45(1), 5–32 (2001)
6. Cochinwala, M., Kurien, V., Lalk, G., Shasha, D.: Efficient data reconciliation. Information Sciences 137(1), 1–15 (2001)
7. Cohen, W.: Integration of heterogeneous databases without common domains using queries based on textual similarity. ACM SIGMOD Record 27, 201–212 (1998)
8. Dai, W., Yang, Q., Xue, G., Yu, Y.: Boosting for transfer learning. In: Proceedings of the 24th International Conference on Machine Learning, pp. 193–200. ACM (2007)
9. Eaton, E., desJardins, M., et al.: Selective transfer between learning tasks using task-based boosting. In: Twenty-Fifth AAAI Conference on Artificial Intelligence (2011)
10. Ell, B., Vrandečić, D., Simperl, E.: Labels in the Web of 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. 162–176. Springer, Heidelberg (2011)
11. Elmagarmid, A., Ipeirotis, P., Verykios, V.: Duplicate record detection: A survey. IEEE Transactions on Knowledge and Data Engineering 19(1), 1–16 (2007)
12. Freund, Y., Schapire, R.: A Desicion-Theoretic Generalization of On-line Learning and an Application to Boosting. In: Vitányi, P.M.B. (ed.) EuroCOLT 1995. LNCS, vol. 904, pp. 23–37. Springer, Heidelberg (1995)
13. Guha, S., Koudas, N., Marathe, A., Srivastava, D.: Merging the results of approximate match operations. In: Proceedings of the Thirtieth International Conference on Very Large Data Bases, vol. 30, pp. 636–647. VLDB Endowment (2004)

14. Hogan, A., Harth, A., Decker, S.: Performing object consolidation on the semantic web data graph (2007)
15. Hogan, A., Polleres, A., Umbrich, J., Zimmermann, A.: Some entities are more equal than others: statistical methods to consolidate linked data. In: 4th International Workshop on New Forms of Reasoning for the Semantic Web: Scalable and Dynamic, NeFoRS 2010 (2010)
16. Hu, W., Chen, J., Cheng, G., Qu, Y.: Objectcoref & falcon-ao: results for oaei 2010. In: Ontology Matching, p. 158 (2010)
17. Hu, W., Chen, J., Qu, Y.: A self-training approach for resolving object coreference on the semantic web. In: Proceedings of the 20th International Conference on World Wide Web, pp. 87–96. ACM (2011)
18. Isele, R., Jentzsch, A., Bizer, C.: Efficient multidimensional blocking for link discovery without losing recall (2011)
19. Li, J., Tang, J., Li, Y., Luo, Q.: Rimom: A dynamic multistrategy ontology alignment framework. IEEE Transactions on Knowledge and Data Engineering 21(8), 1218–1232 (2009)
20. Loh, W.: Classification and regression tree methods. In: Encyclopedia of Statistics in Quality and Reliability (2008)
21. Ngomo, A., Auer, S.: Limes-a time-efficient approach for large-scale link discovery on the web of data. In: Proceedings of IJCAI (2011)
22. Ngomo, A., Lehmann, J., Auer, S., Höffner, K.: Raven–active learning of link specifications. In: Ontology Matching, p. 25 (2011)
23. Niu, X., Rong, S., Zhang, Y., Wang, H.: Zhishi.links results for oaei 2011. In: Ontology Matching, p. 220 (2011)
24. Pan, S., Yang, Q.: A survey on transfer learning. IEEE Transactions on Knowledge and Data Engineering 22(10), 1345–1359 (2010)
25. Pasula, H., Marthi, B., Milch, B., Russell, S., Shpitser, I.: Identity uncertainty and citation matching. In: Proceedings of NIPS 2002 (2002)
26. Rahm, E., Bernstein, P.: A survey of approaches to automatic schema matching. The VLDB Journal 10(4), 334–350 (2001)
27. Raimond, Y., Sutton, C., Sandler, M.: Automatic interlinking of music datasets on the semantic web. In: Proceedings of the 1st Workshop about Linked Data on the Web. Citeseer (2008)
28. Sleeman, J., Finin, T.: Computing foaf co-reference relations with rules and machine learning. In: Proceedings of the Third International Workshop on Social Data on the Web (2010)
29. Sleeman, J., Finin, T.: A machine learning approach to linking foaf instances. In: Spring Symposium on Linked Data Meets AI. AAAI (January 2010)
30. 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)
31. 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)
32. Wang, Z., Zhang, X., Hou, L., Zhao, Y., Li, J., Qi, Y., Tang, J.: Rimom results for oaei 2010. In: Ontology Matching, p. 195 (2010)
33. Winkler, W.: The state of record linkage and current research problems. In: Statistical Research Division, US Census Bureau. Citeseer (1999)
34. Winkler, W.: Overview of record linkage and current research directions. In: Bureau of the Census. Citeseer (2006)

Who Will Follow Whom? Exploiting Semantics for Link Prediction in Attention-Information Networks

Matthew Rowe[1,2], Milan Stankovic[3,4], and Harith Alani[1]

[1] Knowledge Media Institute, The Open University, Milton Keynes, UK
[2] School of Computing and Communications, Lancaster University, Lancaster, UK
[3] Hypios Research, 187 rue du Temple, 75003 Paris, France
[4] Universit Paris-Sorbonne, 28 rue Serpente, 75006 Paris
{m.c.rowe,h.alani}@open.ac.uk, mail@milstan.net

Abstract. Existing approaches for link prediction, in the domain of network science, exploit a network's topology to predict future connections by assessing existing edges and connections, and inducing links given the presence of mutual nodes. Despite the rise in popularity of Attention-Information Networks (i.e. microblogging platforms) and the production of content within such platforms, no existing work has attempted to exploit the semantics of published content when predicting network links. In this paper we present an approach that fills this gap by a) predicting *follower* edges within a directed social network by exploiting concept graphs and thereby significantly outperforming a random baseline and models that rely solely on network topology information, and b) assessing the different behaviour that users exhibit when making followee-addition decisions. This latter contribution exposes latent factors within social networks and the existence of a clear need for topical affinity between users for a follow link to be created.

1 Introduction

Attention-Information Networks, or '*Hybrid Networks*' [10], lie at the intersection of social and information networks, users can *follow* other users and *subscribe* to the content they publish. Romero and Kleinberg [8] describe such directed interpolating networks as enabling users to become information hubs, in essence such users act as real-time sensors by disseminating information about real-world events and publishing information as it becomes available. Given the large uptake of platforms, such as Twitter (31.9 % increase in users in 2011[1]), that are composed of attention-information networks and the increased number of users to choose from, platform users must carefully select the individuals that they wish to *listen* to. Understanding *who will follow whom* and how users base their decisions - i.e. uncovering follower-decision behaviour patterns - has two key benefits: firstly, the dynamics of network growth in attention-information networks could

[1] http://www.emarketer.com/Article.aspx?R=1008879

be understood and therefore the social capital of the networks be predicted; and secondly, understanding how users behave in terms of their follower-decisions would facilitate audience building, a key interest for online marketing and brand managers who are keen to increase their broadcast spectrum.

Constrained attention capability means that users must decide on who they should follow. One would assume that a followee's content must be of interest to the follower, and this has indeed been identified in prior work by Schifanella et al. [9]. We therefore hypothesise that *Following a user is performed when there is a topical affinity between the follower and the followee.* However, to date no work has attempted to explore the differing behaviour that users may exhibit when making follower-decisions. We hypothesise that *Users who do not focus on specific topics do not base their follower decisions on topical information but on social factors*, as so-called **unfocussed** users who publish content about diverse subjects are not interested in subscribing to other users given a particular subject affinity. Further, we also hypothesise that *Users who are more socially connected are driven by social rather than topical factors*, given that users who build up a large followee network are more driven by connecting to people.

To explore these hypotheses we present an approach to predict links between a follower and recommended followees that exploits the semantics of user content, using tags and the concepts they refer to in order to measure the *semantic relatedness* of users. Our contributions are as follows:

- An approach to predict links in attention-information networks that explores *social*, *topical* and *visibility* factors, based on behavioural differences with regards to user types (alluded to in our hypotheses).
- Evaluation using the KDD Cup 2012 dataset from Tencent Weibo[2] that: a) shows significantly better performance than a random baseline and network topology models, and b) identifies a general pattern for follower-behaviour that is driven by topical affinity.

We have structured the paper as follows: Section 2 formulates our link prediction problem and describes recent work within this area. Section 3 describes the dataset used for our experiments. Section 4 details the prediction approach and the features engineered to capture social, topical and visibility dynamics, and Section 5 describes the method for concept disambiguation. Section 6 presents our experiments to identify follower-decision behaviour patterns and observe how users differ, and Section 7 discusses the findings in comparison with recent work. Section 8 finishes the paper with conclusions and plans for future work.

2 Background and Related Work

2.1 Problem Formulation

A social network can be modelled as a graph $G = \langle V, E \rangle$ where V denotes the set of users (nodes) in the social network and E is the set of edges ($\langle u, v \rangle \in E$)

[2] http://t.qq.com/

between nodes. Link prediction is the task of predicting which nodes $u, v \in V$ will form an edge between one another at a future time step. Leden-Nowell and Kleinberg [7] formulated this problem as detecting the changes in a graph between consecutive time steps ($t_0 < t'_0 < t_1 < t'_1$), by using information in $G[t_0, t'_0]$ to predict the edges in $G[t_1, t'_1]$. However, on attention-information networks the mechanisms through which edges, and therefore social links, are created requires that the link prediction problem is altered to account for recommendations - where a constrained set of possible nodes to connect to is considered. The introduction of recommendation features, such as the '*Who to follow*' feature on Twitter, has shifted the problem to a user-centric task such that a user u is provided with a set of recommendations $R(u)$ to connect to where $R(u) \cap \Gamma(u) = \emptyset$ and $\Gamma(u)$ denotes the ego-centric network of u. Therefore the problem we are addressing is the induction of a link function between users given previously provided recommendations: $f : V \times R \rightarrow \{0, 1\}$, where the set of possible mappings is constrained to the recommendation set of each user ($R(u)$).

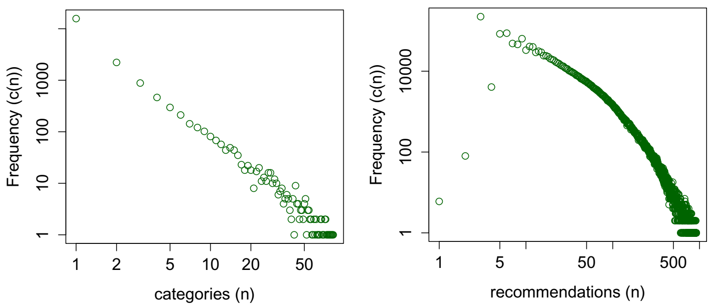
2.2 Related Work

Recent work within the domain of link prediction is divisible into two strands: approaches that use network topologies and approaches that use local metadata. Starting with network topology driven methods, Golder et al. [5] modelled directed paths through networks to assess their effect on follower decisions on Twitter and found that increased transitivity (i.e. directed transitive connections) and common followers was correlated with follower addition. Also experimenting with Twitter, Yin et al. [10] found that 90% of created links are to users within 2 hops of a given user in the social network. Yin et al. assessed path structures through intermediate nodes and derived probabilities based on intermediary connections to predict links. Romero and and Kleinberg [8] examined '*directed closure*' (i.e. directed form of triadic closure) in attention-information networks and found that different link formation behaviour exists between sub-networks. Backstrom and Leskovec [1] proposed a supervised random walks method with restarts that combines node features with edge features to predict future links on Facebook. Edges are equivalent to the affinity between u and v in the context of our work and include features such as common friends. Zhou et al. [12] performed link prediction experiments over a range of network datasets ranging from a protein-protein interaction network through to a co-authorship network, where each network was undirected. The authors found common neighbours between nodes to achieve the best performance.

Focussing now on metadata-driven approaches, Schifanella et al. [9] assessed the correlation between tag affinity (overlap in tag vocabularies) and social neighbours for both Flickr[3] and Last.fm[4] users, finding that users who are close socially have common tags. Similar work by Leroy et al. [6] performed link prediction of Flickr users but with no *a priori* graph information, only using group

[3] http://www.flickr.com

[4] http://www.last.fm



(a) Categories per Tag (b) Recommendations per User

Fig. 1. Distributions in the dataset. Figure 1(a) shows that the distribution of categories per tag and Figure 1(b) shows the distribution of recommendations per user.

membership information to indicate common interests. The authors' approach computed a probabilistic graph in a first *bootstrap* phase before using this information with existing topology-based measures from [7] to boost recall, finding that performance is favourable for common neighbours. Brzozowski and Romero [2] explored the effect of '*homophily*' on user recommendations, measuring this using the Dice coefficient of two users' sets of tags. However, contrary to findings in [9], Brzozowski and Romero [2] found that using similar tags was not useful information for predicting links and instead found mutual followers to be a good predictor. Yin et al. [11] predicted links using random walks with restarts by forming an augmented graph space of person nodes and attribute nodes, where attributes correspond to title keywords in an example DBLP co-authorship dataset, better performance was achieved when using local attributes (i.e. keyword information) rather than existing social connections.

Although existing approaches [9,6,2,11] consider metadata when predicting edges between people the information is constrained to tag sets or group memberships and does not consider concept information. Furthermore, although several works indicate the benefit of using topical information [9,6], there has been no examination of the follower-decision behaviour patterns. In this paper we present an approach that exploits semantics to gauge the topical affinity between a user and potential *followee* using concept graphs, and examine the decision patterns within our induced models.

3 Dataset Description

The dataset that we used for the experiments described later in this paper was the KDD Cup 2012 dataset from the follower prediction task. Participants were given a rich collection of data that included: a) a training set of users with their recommendations, and whether they followed the items or not; b) the following graph of users; c) a set of keywords (tags) found within each user's content, and; d) item-categorisation data. This latter information is what we used for our concept hierarchy as a given item (i.e. a user) is placed within a hierarchical concept graph - e.g. v is placed in 0.1.4.3 where the dot represents the branching of the concept hierarchy - and is also assigned several tags. This data has been manually labelled by the providers of the dataset. Both concept labels and keywords (tags) are anonymised and are replaced by numeric identifiers, so we do not see the underlying data. While the described concept hierarchy is used in our experiments, there is no obstacle to using some other concept hierarchy or structure (e.g. DBPedia) with our approach.

Figure 1(a) shows that many tags appear in a low number of categories, ($\mu = 2.275, \sigma = 5.903$), however certain tags (in the long tail) are extremely ambiguous and appear in many different categories, thus demonstrating the need for concept disambiguation. Figure 1(b) shows the distribution of recommendations per user ($\mu = 29.540, \sigma = 47.492$), with a skew towards lower recommendation counts and only a few users having a large number of recommendations.

4 Predicting Follower-Decisions

In this paper we tackle the problem of predicting links between a user and recommended *followee* users. We formulate the problem as one of inducing a function between the set of users and their recommendations: $f : V \times R \rightarrow \{0, 1\}$. Our goal is to explore various features and: a) identify the best performing general model over all users; and b) explore the decision behaviour of users and the extent to which this differs between them. In order to facilitate accurate predictions and explore the different factors that drive link creation we explore the use of three feature sets: *social*, *topical* and *visibility*. The features contained within these sets are each derived in a pairwise fashion such that if we are provided with a set of recommendations for u denoted by $R(u)$, we measure each feature based on the common information shared over u and $v \in R(u)$.[5]

4.1 Social

The decision of which recommendations to accept may differ between users, for instance it might be the case that one user may only *follow* another with whom they share a mutual friend. In order to assess such dynamics we measure four social features that account for the topology of the network and the existence of edges present within the network prior to predictions.

Mutual Followers Count. Measures the overlap of the follower sets (i.e. the set of users connecting into a given user) between u and v. Let $\Gamma^{-}(u)$ denote the set of followers connected to u and $\Gamma^{-}(v)$ denote the set of users following v, then we define the mutual follower count as:

$$MFR(u, v) = |\Gamma^{-}(u) \cap \Gamma^{-}(v)| \tag{1}$$

[5] We use the symbols u and v hereafter to denote the user and a recommended user respectively.

Mutual Followees Count. Measures the overlap of the followee sets (i.e. the set of users to whom a given user is connected) between u and v. Let $\Gamma^+(u)$ denote the set of followers connected to u and $\Gamma^+(v)$[6] denote the set of users following v, then we define the mutual follower count as:

$$MFE(u,v) = |\Gamma^+(u) \cap \Gamma^+(v)| \tag{2}$$

Mutual Friends Count. Measures the overlap of the friends sets (i.e. the set of users with whom a user is friends, where friendship is denoted by a bi-directional edge between nodes) between u and v. The friend set is derived by taking the intersection of the followee and follower set of a given user. Using this set definition we can then calculate the mutual friends count as the overlap between friend sets between two users, or formally as:

$$MF(u,v) = |(\Gamma^-(u) \cap \Gamma^+(u)) \cap (\Gamma^-(v) \cap \Gamma^+(v))| \tag{3}$$

Mutual Neighbours. Measures the overlap of the ego-centric networks of u and v whilst ignoring the directions of the links in the networks - this measure is taken from [12,10,1]. This feature is included to assess the impact, or lack of, that direction has on link creation - i.e. following or followed. We define this measure formally as:

$$MN(u,v) = |(\Gamma^-(u) \cup \Gamma^+(u)) \cap (\Gamma^-(v) \cup \Gamma^+(v))| \tag{4}$$

4.2 Topical

For certain users the decision to *follow* a user may be based on the content that the other user shares and produces. This effect is symptomatic of *attention-information networks* [5] in which the level of attention that a user can pay to content published by their network members is limited. To explore the effects of topical information on follower decisions we explore the overlap between users in terms of keywords (tags) and concepts. In the following section we describe a method to align a keyword to a concept given a user's context. Given such concepts we can explore the relation between users in terms of their semantic distance from one another within a concept graph, the intuition being that the further away two users are, then the less similar they are in terms of their interests, allowing the effect of *homophily* to be explored. We define several conventions as follows: let T_u be the set of keywords (or tags) found within the content of user u and C_{T_u} be the bag of concepts attributed to the tags from T_u.

Tag Vectors - Cosine Similarity. Our first feature is similar to the cosine similarity between user tag vocabularies described in [9]. We define the tag vector $\boldsymbol{t_u} = \{t_1, t_2, ..., t_n\}$ of a user u as being derived from the user's tag set T_u using a binary index of the appearance of a tag within a user's content - i.e. $t_. = \{0, 1\}$ in

[6] We use the symbols $-$ and $+$ in the superscripts of the ego-centric networks to denote the direction of the edges, the former denoting incoming and the latter denoting outgoing.

$\boldsymbol{t_u}$. To compute the similarity between the tag vectors of u and v denoted by $\boldsymbol{t_u}$ and $\boldsymbol{t_v}$ respectively we take the cosine of the angle between these vectors.

Concept Bags. The concept bag C_{T_u} of a given user u is derived by returning the set of concepts that each tag in T_u has been associated with. As we have a collection of tags it is likely that duplicate concepts will be returned for different tags, we maintain these duplicates in the *concept bag* of a user and form a *concept bag vector*: $\boldsymbol{c_u} = \{c_1, c_2, ..., c_n\}$ using the frequency of the concepts in the bag as the weights - i.e. $c_. = \{0\} \cup N^+$ in $\boldsymbol{c_u}$.

Cosine Similarity. Given two concept bag vectors $\boldsymbol{c_u}$ and $\boldsymbol{c_v}$ for two different users u and v respectively, we measure the cosine of the angle between those vectors as the first measure between concept bags.

Jensen-Shannon Divergence. Given two concept bag vectors $\boldsymbol{c_u}$ and $\boldsymbol{c_v}$ for two different users we model each vector as a probability distribution over the total set of concepts denoted as P_u and P_v respectively - using frequency counts for keyword usages to derive the probability distributions. Using these distributions over the total set of concepts we then measure the Jensen-Shannon Divergence between the concept bags, thereby gauging the level of dissimilarity between the concepts attributed to the content of u and v:

$$D_{JS} = \frac{1}{2}\sum_i P_u(i) \log \frac{P_u(i)}{P_v(i)} + \frac{1}{2}\sum_i P_v(i) \log \frac{P_v(i)}{P_u(i)} \tag{5}$$

Concept Graphs. By using concept graphs we can explore the semantic relatedness of users using graph-based distance metrics. To enable this comparison we require a one-to-one mapping between a tag and a concept given a user. This produces a set of concepts for each user that can be used for comparison with other users. In the following section we explain how we perform *concept disambiguation* using a user's context to overcome the polysemy problem - i.e. where a single tag can have multiple concepts. We define $\langle t, c\rangle \in M_u$ as an injective map between the tags from the tag set T_u of user u and the set of concepts from the concept bag C_{T_u} of user u where a concept aligned to a tag given a user is returned by $M_u[t] = c$. Through this we can perform a pairwise comparison of the distances between concepts attributed to the tags of u and v using the function: $d(c_i, c_j)$. We define three distance measures over concept graphs - these distance measures are explained shortly - each of which have two varieties:

1. *Tag Intersection.* The first variety uses the intersection of the tag sets of u and v for comparison: $T_u \cap T_v = T_{uv}$. For each tag in $t \in T_{uv}$ we produce the tag-concept maps given each user such that $|M_u| \equiv |M_v|$. The distances between the concepts from equivalent tags in T_{uv} are measured using a distance metric and the average taken. We define this formally as:

$$INT(T_{uv}) = \frac{1}{|T_{uv}|} \sum_i^{|T_{uv}|} d(M_u[t_i], M_v[t_i]) \tag{6}$$

2. *All Tags.* The second variety performs a pairwise distance comparison between the tag sets of u and v through the concepts that each tag within those sets has been mapped to. For each tag in a given user's tag set T_u we produce an injective map: M_u. However, unlike the tag intersection the cardinality of the map for one user will differ from another if the cardinality of their tag sets differs. Using the maps we then measure the distance between every mapped concept in the different sets. We define this formally as:

$$ALL(T_u, T_v) = \frac{1}{|T_u|} \frac{1}{|T_v|} \sum_{i}^{|T_u} \sum_{j}^{|T_v|} d(M_u[t_i], M_v[t_j]) \tag{7}$$

Based on these two varieties we explore three distance metrics for measuring $d(c_i, c_j)$ in the concept graph:

Shortest Path. The first metric derives the shortest path between c_i and c_j using the Bellman-Ford algorithm. This method performs a breadth first search of a graph-space until a desired node is found.

Hitting Time. The second and third metrics utilise the Markov-chain random walks model in which the probability of a random walker moving from one node to another in one time step is only dependent on their current position in a graph. The graph over which the random walker will traverse is the concept graph G_{conc} which is composed of nodes (concepts) V_{conc} and edges that connect those concepts $\langle i, j \rangle \in E_{conc}$ - where edges are undirected and therefore hypernym and hyponym relations are ignored for now. We define the random walks model using the Laplacian matrix of the concept graph: $\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{A}$ and define the adjacency matrix $\boldsymbol{A}$ for entry a_{ij} to be 1 if $\langle i, j \rangle \in E_{conc}$ and 0 otherwise.

The diagonal degree matrix is defined as the row sum of the adjacency matrix: $d_{ii} = \sum_j a_{ij}$. We then take the Moore-Penrose pseudoinverse of the laplacian matrix which we denote as L^+. This provides, based on work by Fouss et al. [4], the necessary information to efficiently derive the hitting time $m(j|i)$ of a random walker as it traverses the concept graph G_{conc}, this is computed as follows:

$$m(j|i) = \sum_{k}^{|V_{conc}|} (l^+_{ij} - l^+_{ik} - l^+_{jk} + l^+_{kk}) d_{ii} \tag{8}$$

Commute Time Distance. The third distance metric computes the average number of steps the walker takes to leave a given node i reach another node j and then return back to i. The closer that two nodes are in the concept graph G_{conc} then the shorter the commute time. As the hitting time distances are not symmetric - i.e. $m(j|i) \neq m(i|j)$ - we define the commute time distance from i to j as: $n(j|i) = m(j|i) + m(i|j)$.

4.3 Visibility

The presence and access to information published by a prospective followee could influence users in deciding whether to follow the individual or not. However, limitations imposed on attention-information networks means that the dominant, but not solitary, method through which posts by individuals outside of a user's followee network are seen is if they are *retweeted* or if a followee *mentions* a user. To explore these effects we devised the following six features:

Retweet Count. The total number of times a given user (v) has been retweeted by members of the followee network belonging to u ($w \in \Gamma^+(u)$), we define this as follows, using the $retweet(w,v)$ function to return the number of times w retweeted v:

$$RC(u,v) = \sum_{w \in \Gamma^+(u)} retweet(w,v) \tag{9}$$

Mention Count. The total number of times a given user (v) has been mentioned by members of the followee network belonging to u ($w \in \Gamma^+(u)$), we define this as follows, using the $mention(w,v)$ function to return the number of times w mentioned v:

$$MC(u,v) = \sum_{w \in \Gamma^+(u)} mention(w,v) \tag{10}$$

Comment Count. The total number of times a given user (v) has had his/her content commented on by members of the followee network belonging to u ($w \in \Gamma^+(u)$), we define this as follows, using the $comment(w,v)$ function to return the number of times w commented on content published by v:

$$CC(u,v) = \sum_{w \in \Gamma^+(u)} comment(w,v) \tag{11}$$

Weighted Retweet Count. The retweet count gauges the number of times a user v has been retweeted by members of the followee network $\Gamma^+(u)$ of u. The influence that members of this followee network exhibit may differ depending on the attention that u pays to each person. To assess this effect we set δ_w to be the number of times u has replied to $w \in \Gamma^+(u)$. We then derive a normalised influence weight λ_w for $w \in \Gamma^+(u)$ such that $\sum_{w \in \Gamma^+(u)} \lambda_w = 1$, where $\lambda_w = \delta_w / \sum_{w_{.} \in \Gamma^+(u)} \delta_{w_{.}}$. Given this influence weighting scheme we then measure the weighted retweet count such that the neighbours of u assert different effects on the count:

$$WRC(u,v) = \sum_{w \in \Gamma^+(u)} \lambda_w.retweet(w,v) \tag{12}$$

Weighted Mention Count. As above, with the weighted retweet count, we also adjust the mention count of v by members of the followee network of u based on attention:

$$WMC(u,v) = \sum_{w \in \Gamma^+(u)} \lambda_w.mention(w,v) \tag{13}$$

Weighted Comment Count. The comment count is also adjusted based on the attention paid by u to his followee network members:

$$WCC(u,v) = \sum_{w \in \Gamma^+(u)} \lambda_w.comment(w,v) \tag{14}$$

5 Concept Disambiguation with User Contexts

Measuring the distances between concepts within a graph space provides a notion of *semantic relatedness* that can, in turn, be used to cumulatively gauge the topical similarity between two users. As we mentioned above, distances are measured using three different metrics, however each metric requires an injective map between a set of tags and the concepts that they refer to. In this context we encounter the problem of concept ambiguity, also known as *polysemy*, where a single tag can have multiples concepts mapped to it.[7] Our earlier assessment of the distribution of categories per tags, as shown in Figure 1(a), demonstrates the large extent to which polysemy is evident within the dataset.

Cantador et al. [3] proposed '*distributional aggregation*' as a method for choosing the most representative tag for a web resource based on usage frequency amongst a collection of users. Our approach performs concept disambiguation by leveraging the context of the user, thereby swapping the collection of users for the concept bag of a given user and exploiting that as a *voting* mechanism. To illustrate this better consider a scenario in which the tag sets T_u and T_v and concept bags C_{T_u} and C_{T_v} are returned for for u and v. For each tag in the tag set we derive the list of candidates $C_{cand,t}$ for that tag (t) from the concept graph. For instance for a tag $t1$ we may have two candidates in the candidate set: $\{c1, c2\} \in C_{cand,t1}$. We count how many times each candidate appears in the concept bag of the user C_{T_u} and choose the most frequent, this then forms the mapping for the tag: e.g. $M_u[t1] = c1$. We define this process using the following function:

$$CD(C_{cand,t}, C_{T_u}) = \arg\max_{c} |\{c : c \in C_{cand,t}; c \in C_{T_u}\}| \tag{15}$$

6 Experiments

In the introduction of this paper we stated three hypotheses that describe the follower-decision behaviour of members of attention-information networks. The aforementioned features are engineered to capture the social, topical and visibility factors that could lead to follower decisions. In this section we describe experiments to verify our hypotheses tested over the KDD Cup 2012 dataset.

6.1 Experimental Setup

Our task, given that we are inducing a link function ($f : V \times R \rightarrow \{0, 1\}$), is a binary classification one. In essence we are asking *will user u follow user v?* To test our hypotheses we performed two experiments: *General Follower Prediction* and *Binned Follower Prediction*. For each experiment we first performed *model selection* by inducing a logistic regression model using only social, topical or visibility features and then all features combined together, we then selected the best model by the one that maximised the Area Under the ROC Curve (*AUC*).

[7] This is analogous to a word having multiple senses on Wordnet or the same tag appearing in multiple fixed taxonomical categories.

Second we assessed the coefficients in the logistic regression model trained on *all* features to identify patterns between an increase in a feature and the log-odds of the classifier increasing, and therefore the likelihood of a follower decision also increasing. The experiments were setup as follows:

General Follower Prediction. Our first experiment sought a general follower prediction model in order to observe differences, at a general level, in follower behaviour. We randomly selected 10% of users from the dataset and generated a machine learning dataset for each user by building feature vectors $\boldsymbol{x_v}$ that contained the social, topical and visibility features (19 features in total) computed in a pairwise fashion between u and each recommended user $v \in R(u)$, setting the class label to *pos* if u followed v or *neg* otherwise. We combined the user-specific datasets together into one large dataset and balanced the data such that there were an equal number of positive and negative examples. The dataset, following balancing and a further randomisation process to ensure mixing, was then divided into an 80:20% split for training and testing, containing 457,722 instances in total.

Binned Follower Prediction. We performed two experiments in this context. To begin with we measured two metrics for each user:

1. *Concept-bag Entropy:* We took the concept bag of each user derived from their tag set and measured the entropy of that concept bag, thereby capturing the dispersion of concepts that the user could be talking about. In this context **low entropy** denotes a **focussed** user while **high entropy** denotes an **unfocussed** user who is more random in the subjects that he publishes. We define this measure as follows, where $p(c_j)$ is the conditional probability of the concept (c_j) within the user's concept bag (C_{T_u}):

$$H_{C_{T_u}} = - \sum_{j=1}^{|C_{T_u}|} p(c_j) \log p(c_j) \tag{16}$$

2. *Degree Distribution:* We measure this as the proportion of users on the platform the user follows, thereby gauging how connected a user is. To derive this measure we took the out-degree of each user $(|\Gamma^{+}(v)|)$ and divided this by the total number of users $(|V|)$.

For each measure we divided the users up into 10 equal-frequency bins such that the same number of users were placed within each bin and selected all the users from the *low* and *high* bins. By choosing 10 bins and then selecting the low and high users for each of the above measures we are provided with users who will differ greatly in these properties. Following this binning process we built the datasets for each binned user using the same process as above (i.e. building pairwise feature vectors for each user u and each of his recommended users $v \in R(u)$) and then combined the user-specific datasets together, thereby producing four datasets for the experiment: two for the *Concept-bag Entropy* (*low* with 268,818 and *high* with 325,508 instances) and two for the *Degree Distribution* (*low* with 400,866 and *high* with 610,098 instances). We balanced each dataset such that there were the same number of positive and negative instances and then divided each dataset up into a training/testing sets using an 80:20 split.

Evaluation Measures. To assess the accuracy of the trained logistic regression models we measured the area under the Receiver Operator Characteristic Curve (AUC).[8] We use this measure to choose a model that predicts links accurately and minimises the number of false predictions. We also computed the Matthews correlation coefficient (MCC) to compare our models against a random predictor baseline: a coefficient of $+1$ is a perfect prediction, 0 is equal to random and -1 is total disagreement between prediction and observation.

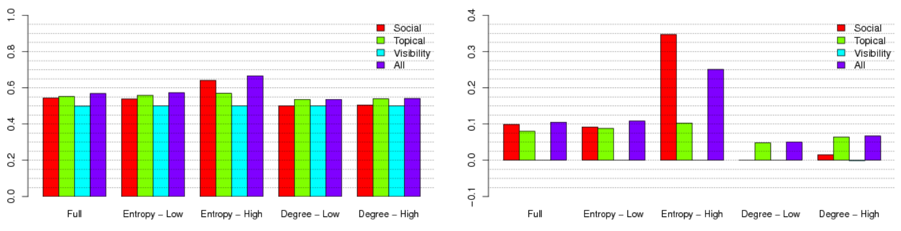
6.2 Results: Prediction Accuracy

We begin our analysis of the prediction models by assessing the accuracy levels achieved in each experiment, as shown in Figure 2. Starting with the full model and assessing the feature sets used in isolation the results indicate that **topical** factors achieve the best performance and provide the most useful information for predicting links between users - performing significantly better (t-test with $\alpha < 0.001$) than social and visibility features. Within the introduction of this paper we hypothesised that '*Following a user is performed when there is a topical affinity between the follower and the followee*', the findings from our assessments of feature sets used in isolation confirms this hypothesis, however combining all the features together achieves the best performance. To provide an indication of the difference between the performance of the model and the baseline we ran the sign test of the MCC values and found each feature set combination to significantly outperform the random model.

Inspecting the AUC values produced for the Concept-bag Entropy models we find different patterns. For the *low entropy* users the **topical** factors perform best of the isolated feature sets, in a similar manner to the *full* model, while for the *high entropy* users the **social** features achieve the best performance of the isolated sets - significantly better than the other models (t-test with $\alpha < 0.001$). This finding confirms our earlier hypothesis that '*Users who do not focus on specific topics do not base their follower decisions on topical information but on social factors*'. Our intuition was that the more random a user is in his discussions then the less likely it would be for that user to base his follower-decisions on topical information. Instead, the driver in making such a decision is more likely to be social, as the user is more inclined to spread the topics and subjects he discusses in order to engage with more people.

Turning now to the Degree Distribution models we also find similar results to the full model by achieving the highest AUC values when using the **topical** features. Interestingly, we hypothesised earlier that '*Users who are more socially connected are driven by social rather than topical factors*', yet Figure 2 indicates that **topical** features outperform **social** features, thereby rejecting our earlier hypothesis - the former features were found to be significantly better (t-test with $\alpha < 0.01$). It could be the case that users who have a high out-degree form topic

[8] This accuracy measure is used throughout link prediction literature [12,9,6].



(a) *AUC* (b) *MCC*

Fig. 2. Follower Prediction Model Results using the Full model and the Binned models. We report the Area under the ROC curve (*AUC*) and the Matthews Correlation Coefficient in parentheses, significantly outperforming the random baseline for all models bar **visibility** features.

specific communities, assessing the coefficients of the logistic regression model for these high degree users should confirm this. We find that for all the models **visibility** features have little effect on predictions as only a small minority are non-zero - i.e. Retweet Count: $\mu = 13 \times 10^{-5}$.

6.3 Results: Follower-Decision Patterns

We now study the patterns of the logistic regression models in greater detail to compare the effects of various features on the log-odds ratio of the classifier and the probability of a user creating a link in their followee network. Starting again with the *full* model, and assessing the coefficients in Table 1,[9] we find that connections are formed between two users when there are fewer mutual followers, but more mutual neighbours - so they share some social affinity through their neighbours. Topically, users follow other users who are closer to them in terms of the subjects they discuss characterised by the reduced JS-divergence and greater cosine similarity across the tags and concepts, and also the reduced shortest path and hitting between *all* concepts of the users.

In terms of the binned models: *low entropy* users follow other users with whom they share less mutual followers but more mutual neighbours. These users should also have a greater topical affinity, given the negative coefficients for JS-divergence and the shortest path and hitting time across all concepts, and the positive coefficients for the tag vector and concept bag cosines. While *high entropy* users (who cover a lot of different topics in their discussions) follow other users with whom they share more mutual followers but less mutual friends. We also observe an interesting behaviour pattern for these user types as the tag vector cosine between a user and his followee should be minimised - indicating the requirement for a reduced overlap in the keywords that both users publish - however the concept vector cosine should be increased and the JS-divergence and hitting time should be reduced. As these latter features cover concept information, abstracted from tags published by either user, this suggests the presence of topical affinity between users without either user talking about the same tags.

[9] We only comment on features whose inclusion in the model is significant.

Table 1. Follower Prediction Model Coefficients for the General model (full) and the Binned models (Concept-bag Entropy and Degree Distribution)

		Full	Concept-bag Entropy		Degree Distribution	
Set	Feature		Low	High	Low	High
Social	Mutual Followers Count	-0.0275***	-0.0497***	0.2985***	8.6776	-0.0062**
	Mutual Followees Count	0.0001	-0.0064	0.1440***	-	0.0066***
	Mutual Friends Count	-0.0236	-0.1357	-0.2786***	-	0.0041
	Mutual Neighbours Count	0.0289***	0.0462***	-0.3033***	-	0.0023.
Topical	Tag Vectors - Cosine	0.7887***	0.5793**	-0.5125***	0.8628***	0.4840**
	Conc Bags - Cosine	0.6277***	0.9587***	1.6519***	0.5624***	0.5779***
	Conc Bags - JS-Divergence	-0.0410***	-0.0421**	-0.6369***	-0.0425***	-0.0059
	Conc Graphs - Int - Short Path	0.0329.	0.0811***	0.1324***	0.0556*	0.0795***
	Conc Graphs - All - Short Path	-0.0659***	-0.0444***	0.1515***	-0.0516***	-0.1230***
	Conc Graphs - Int - Hit Time	0.0009***	-0.0001	-0.0003**	-0.0002	-0.0002
	Conc Graphs - All - Hit Time	-0.0007***	-0.0006***	-0.0001.	-0.0005***	-0.0004***
	Conc Graphs - Int - Com Time	-0.0005***	0	0.0001	0.0001	0
	Conc Graphs - All - Com Time	0.0004***	0.0003***	0	0.0003***	0.0003***
Visibility	Retweet Count	4.3102	8.2279	6.9181	-	0.4570
	Mention Count	-12.5017	-	-	-	-2.4563
	Comment Count	-8.4571	-	-	-	-2.1373
	Weighted Retweet Count	-	-	-	-	-
	Weighted Mention Count	-	-	-	-	-
	Weighted Comment Count	-20.2386	-381.3810	-1401.6106	-	-1.1584

Signif. codes: p-value < 0.001 *** 0.01 ** 0.05 * 0.1 . 1

For *low degree* users we find that largely topical features appear within the model - as these users are not connected to many people and therefore the appearance of social factors is diminished. For these users the coefficients indicate a similar pattern for *low entropy* users where a user follows a recommended user if they share topical affinity - i.e. high cosine similarity based on tags and concepts, and lower hitting time and JS-divergence. *High degree* users follow other users with whom they share fewer mutual followers but more mutual followees - i.e. both users need to follow many of the same people. For topical features the high degree users follow other users with whom they share a topical affinity, indicating that although these users *subscribe* to many users they are based on common subjects and interest. We also find similar topical effects to both the *full* model and the *low entropy/degree* models: high similarity between the follower and followee based on tag vector/concept bag cosine, and low JS-divergence and hitting times.

7 Discussion

Analysing the follower-decision behaviour of users in an example microblogging platform proved two of our three earlier stated hypotheses. We also uncovered a general behaviour pattern based on the topical affinity between a follower and followee where the concept distance between the users should be lowered - measured using the random walks hitting time and tag similarity. Such findings are consistent with work by Schifanella et al. [9] where users who were socially close to one another were found to have a high topical affinity (cosine of tag vocabularies). Our work attempts to advance such findings by exploring novel metrics for assessing topical affinity, using concept graphs, and inspecting the coefficients in induced logistic regression models to unearth the latent behavioural pattern. Such an examination has never been undertaken before.

In comparison with existing work we found different follower-decision behaviours. For instance, Golder et al. [5] found that on Twitter common followers, when increased in number, boosted the likelihood of a link being formed. This was also reported in Brzozowski and Romero [2] where sharing a mutual audience was correlated with better link prediction. However we do not see this effect in our general model and only see the increase in mutual followers as increasing the likelihood of a user following another in the *high entropy* model. In fact for the remaining models, the number of mutual followers should actually be reduced, thereby conflicting with both Golder et al. and Brzozowski and Romero's findings. Leroy et al. [6] found that an increase in mutual neighbours between a follower and followee was correlated with edge creation. We also observe a similar effect in our models where for all models, aside from the *high entropy* users, an increase in the number of mutual neighbours was associated with an increase in link creation likelihood. This divergent behaviour for *high entropy* users is common across many of the implemented features and suggests the need for model adaptation when considering these user types. Despite such divergent behaviour we note that a consistent topical affinity effect exists, as conceptually - i.e. considering concepts abstracted from published keywords - we find topical affinity between such random users and their followees.

8 Conclusions and Future Work

In this paper we have presented an approach to predict links on attention-information networks and in doing so: a) significantly out-performed a random model baseline when using all implemented features in a logistic regression model and existing topological models when using topical information; b) learnt a general pattern that captures the follower-behaviour of users of an example microblogging platform; and c) uncovered latent factors that lead to link creation including clear topical affinity between followers and followees. These findings allow followee recommendations to be improved based on the behaviour of the recipient, and therefore grow the network on the platform and increase social capital. A necessary next step for this work is to apply our models over data from other attention-information networks such as Twitter and YouTube in order examine the behaviour of their users and whether the findings from this work corroborate with those from disparate platforms. Assuming that concepts resolvable to Linked Data URIs can be extracted from textual content available in those networks, this would allow our concept-based affinity measures to be applied over Linked Data.

Future work will also involve assessing the correlation between social network distances between users and their topical affinity. Schifanella et al. [9] found when one compares users who are more than 2 steps away in a social network then the topical affinity between users declines rapidly. We plan to combine this examination with applying our approach over Twitter and YouTube. We are also exploring the use of Conditional Random Fields on link prediction, thereby allowing follower-decisions to be conditioned on recent user behaviour - i.e. a user's recent propensity to follow other users. We conjecture that performance is conditioned on time-sensitive behaviour of each user. We do not capture this at present.

Acknowledgments. This work was supported by EU-FP7 project ROBUST (grant no. 257859).

References

1. Backstrom, L., Leskovec, J.: Supervised random walks: predicting and recommending links in social networks. In: Proc. of the International Conference on Web Search and Data Mining, WSDM 2011. ACM, New York (2011)
2. Brzozowski, M.J., Romero, D.M.: Who Should I Follow? Recommending People in Directed Social Networks. In: International AAAI Conference on Weblogs and Social Media (2011)
3. Cantador, I., Bellogín, A., Fernández-Tobías, I., López-Hernández, S.: Semantic Contextualisation of Social Tag-Based Profiles and Item Recommendations. In: Huemer, C., Setzer, T. (eds.) EC-Web 2011. LNBIP, vol. 85, pp. 101–113. Springer, Heidelberg (2011)
4. Fouss, F., Pirotte, A., Renders, J.-M., Saerens, M.: Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation. IEEE Trans. on Knowl. and Data Eng. (March 2007)
5. Golder, S.A., Yardi, S.: Structural predictors of tie formation in twitter: Transitivity and mutuality. In: Proc. of the IEEE International Conference on Social Computing, Washington, DC, USA (2010)
6. Leroy, V., Barla Cambazoglu, B., Bonchi, F.: Cold start link prediction. In: Proc. of the International Conference on Knowledge Discovery and Data Mining. ACM, New York (2010)
7. Liben-Nowell, D., Kleinberg, J.: The link-prediction problem for social networks. J. Am. Soc. Inf. Sci. Technol. 58(7), 1019–1031 (2007)
8. Romero, D.M., Kleinberg, J.M.: The directed closure process in hybrid social-information networks, with an analysis of link formation on twitter. In: ICWSM (2010)
9. Schifanella, R., Barrat, A., Cattuto, C., Markines, B., Menczer, F.: Folks in folksonomies: social link prediction from shared metadata. In: Proce. of the International Conference on Web Search and Data Mining, New York, NY, USA (2010)
10. Yin, D., Hong, L., Davison, B.D.: Structural link analysis and prediction in microblogs. In: Proc. of the ACM International Conference on Information and Knowledge Management, New York, NY, USA (2011)
11. Yin, Z., Gupta, M., Weninger, T., Han, J.: Linkrec: A unified framework for link recommendation with user attributes and graph structure. In: Proc. of the World Wide Web Conference, New York, NY, USA (2010)
12. Zhou, T., Lü, L., Zhang, Y.-C.: Predicting missing links via local information. The European Physical Journal (October 2009)

On the Diversity and Availability of Temporal Information in Linked Open Data

Anisa Rula[1], Matteo Palmonari[1], Andreas Harth[2], Steffen Stadtmüller[2], and Andrea Maurino[1]

[1] University of Milano-Bicocca
{rula,palmonari,maurino}@disco.unimib.it
[2] Karlsruhe Institute of Technology (KIT)
{harth,Steffen.Stadtmueller}@kit.edu

Abstract. An increasing amount of data is published and consumed on the Web according to the Linked Data paradigm. In consideration of both publishers and consumers, the temporal dimension of data is important. In this paper we investigate the characterisation and availability of temporal information in Linked Data at large scale. Based on an abstract definition of temporal information we conduct experiments to evaluate the availability of such information using the data from the 2011 Billion Triple Challenge (BTC) dataset. Focusing in particular on the representation of temporal meta-information, i.e., temporal information associated with RDF statements and graphs, we investigate the approaches proposed in the literature, performing both a quantitative and a qualitative analysis and proposing guidelines for data consumers and publishers. Our experiments show that the amount of temporal information available in the LOD cloud is still very small; several different models have been used on different datasets, with a prevalence of approaches based on the annotation of RDF documents.

Keywords: temporal information, temporal annotation, linked data.

1 Introduction

The problem of managing temporal information has been deeply studied in the field of temporal databases [18] and has been more recently addressed in the World Wide Web domain [9,1]. In fact, most data-driven and Web applications need to manage temporal information in order to capture, model, explore, retrieve, and summarize information changing over time. Moreover, the amount of rapidly changing data is likely to grow in the next future with the increasing publication of sensor data, which explicitly represents real-time data of evolving phenomenon over time [19,25,27]. As the information on the Web can change rapidly [4], also Linked Data on the Web[1] cannot be assumed to be static, with RDF statements frequently added to and removed from published datasets [29].

[1] http://lod-cloud.net/

As a consequence, change management and temporal information are receiving an increasing attention in the Linked Data domain. In particular, a number of significant issues have been investigated: a resource versioning mechanism for Linked Data, which allows for publishing time-series of descriptions changing over time [7]; a method to monitor the published datasets, successfully applied to several sources [17]; the maintenance of links over evolving datasets [24].

The capability of managing temporal information plays also a crucial role in several applications and research areas. In *Semantic Data Integration*, temporal information can be used to favor the most up-to-date information when fusing data [22,23]. The analysis of temporal information can also support entity resolution in some complex scenarios where the values of the attributes considered in the matching process change over time [21]. In *Temporal Query Answering and Search*, temporal information can be used to filter out the data of interest given some temporal constraint, or to rank the results of a search engine on a temporal basis. Timelines associated with data can improve the *User Experience* by presenting information in a time-dependent order [30,1].

The capability of designing effective solutions depends on the availability of temporal information and the possibility to collect and process this information across heterogeneous datasets. For example, the modification date associated with RDF documents and extracted via HTTP protocol analysis has been used to fuse data coming from different DBpedia datasets [22]; however, this information is not available in many datasets. Understanding the current status of temporal information published as Linked Data is fundamental for the development of applications able to deal with the dynamism in the data.

In this paper we investigate temporal information published in Linked Data on the Web by analysing its availability and characterisation both from a quantitative and qualitative perspective. To the best of our knowledge, despite the proposal of several approaches to model and query temporal information in RDF [11,5,30,19], support versioning for Linked Data [24], and monitor changes [29,17], a systematic and large scale analysis in this field is still missing. Based on a more precise definition of the concept of *temporal information*, we identify a specific kind of temporal information, called *temporal meta-information* in the paper. Temporal meta-information is particularly relevant to several application domains because it associates RDF statements and graphs with information about their creation, modification and validity. Since the analysis of the whole LOD cloud is unfeasible, we use the large Billion Triple Challenge[2] (BTC) dataset for our investigation. In particular, we focus on the characterization and availability of temporal meta-information, reviewing the proposed models in the literature for modelling such information and analysing their usage in the BTC.

The analysis of the BTC corpus suggests that the availability of temporal information is still scarce, with negative consequences on the design of effective solutions leveraging temporal information at large scale. Moreover, we found that none of the models proposed to manage temporal information has been widely adopted, although temporal annotations of documents seem to prevail so

[2] http://km.aifb.kit.edu/projects/btc-2011/

far. Based on the results of our empirical analysis, we provide some guidelines to data publishers and consumers in order to take advantage of the representation approaches proposed so far.

The paper is organized as follows: Section 2 introduces the preliminary definitions we adopt in this paper; in Section 3 we introduce the notion of temporal information and we investigate their availability in the BTC, analysing the more frequent temporal properties and the pay-level-domain they occur in. In Section 4, we review the approaches proposed in the literature for the representation of temporal meta-information and discuss their adoption in well-known datasets. In Section 5 we conduct experiments to quantitatively investigate the adoption of these models in the LOD cloud using the BTC dataset and we discuss our findings. In section 6, we draw the conclusions.

2 Preliminaries

RDF triples and RDF graphs. Given an infinite set $\mathcal{U}$ of URIs (resource identifiers), an infinite set $\mathcal{B}$ of blank nodes, and an infinite set $\mathcal{L}$ of literals, a triple $\langle s, p, o\rangle \in (\mathcal{U} \cup \mathcal{B}) \times \mathcal{U} \times (\mathcal{U} \cup \mathcal{B} \cup \mathcal{L})$ is called an *RDF triple*; s, p, o are called, respectively, the subject, the predicate and the object of the triple. An *RDF graph* G is a set of RDF triples. A *named graph* is a pair $\langle G, u\rangle$, where G is a graph and $u \in \mathcal{U}$. RDF data are often stored using the N-quad format; a quad is a quadruple $\langle s, p, o, c\rangle$ where c defines the context of an RDF triple $\langle s, p, o\rangle$; the context describes the provenance of a triple, often represented by - but not limited to - an RDF graph. An RDF triple (or simply triple in the following) is also called *statement*. We call statements and graphs also *truth-valuable RDF elements*, as they can be associated with a truth value, under an interpretation function [10].

Temporal entities. We distinguish two types of temporal entities used for representing temporal information in RDF data: *time points*, represented by a single variable t^p, and *time intervals*, represented by the standard notation $[t^b; t^e]$, where t^b and t^e represent the time points respectively beginning and ending the interval and $t^b \leq t^e$ (in this paper we do not consider representations of time where intervals are not bound by time points).

Concrete Representation of Time Points on the Web. According to well-accepted best practices, time points are represented on the Web by means of *date formats.* RFC 2616 defines three different date formats that are used in the HTTP protocol[3]. The first *datetime* format, e.g., `Sun, 07 Sep 2007 08:49:37 GMT`, is defined by the standard RFC 822 [6] and is the most preferred. The second *datetime* format, e.g., `Sunday, 07-Sep-07 08:49:37 GMT`, is defined by the standard RFC 850 [15]. The third datetime format, e.g., `Sun Sep 7 08:49:37 2007`, is defined by ANSI C's *asctime* format. ISO 8601 defines a numerical date format [16]; an example of date according to this format is 2007-09-07T08:49:37.sZ. Based on this standard, dates can be also modelled as primitive datatypes in XML Schema [8]. The primitive types, `date`, `dateTime`,

[3] `http://www.ietf.org/rfc/rfc2616.txt`

`gYearMonth`, `gYear`, `gMonthDay`, `gDay` and `gMonth` defined by these specifications are usually used in RDF data. An alternative representation of time for Linked Data, which denotes temporal entities with URIs and makes use of the OWL Time ontology [12] has also been proposed [5].

RDF statements and documents. Some URIs occurring in RDF statements denote resources that are, in fact, documents (e.g., XML documents, PDF documents, or HTML pages). For the purpose of this paper it is relevant to distinguish between *generic documents* and documents publishing RDF data, called *RDF documents* in the following; like other generic documents, RDF documents can be described *by* RDF descriptions, but differently from other documents, they also contain truth-valuable RDF elements (statements and graphs). In other words, a description about an RDF document can provide a meta-description about the content of the RDF document[4].

3 Temporal Information and Temporal Properties

In this section, we first propose an abstract definition of temporal information by introducing the concept of temporal meta-information. Then we analyse the availability of temporal information in Linked Data and the properties that are used more often to represent such information.

- **Temporal information.** At the abstract level a *temporal information* can be described as a ternary relation $T(x, a, t)$, where x is a resource, a statement, or a graph, a is a property symbol, and t is a temporal entity. We call *temporal property* any property symbol used in a temporal information. Since a temporal information $T(x, a, t)$ can be also interpreted as a temporal annotation associated with the element x, the terms temporal information and temporal annotation will be used interchangeably, depending on the context.
- **Temporal meta-information.** We observe that, according to the above definition, truth valuable and non truth valuable RDF entities can be associated with temporal information. Therefore, we introduce a new concept that specifically refers to temporal information associated with truth-valuable elements: a temporal information $T(x, a, t)$ is a *temporal meta-information* if and only if x is a truth-valuable RDF element. The concept of temporal meta-information, which is defined according to semantic criteria, allows distinguishing between temporal information associated with objects in a domain of interest (e.g. the birth date of a person, but also the creation date of a PDF document) and temporal information associated with truth-valuable RDF elements (e.g, the temporal validity of statement, or the last update of an RDF document).

[4] An increasing number of RDF descriptions are also available in the RDFa syntax from plain HTML and XHTML documents; however, in this paper we focus only descriptions available in RDF/XML documents because the crawled data of the BTC corpus, which we use in our analysis, do not include data extracted from RDFa sources.

3.1 Dataset and Experimental Setup

To give more insights about the usage of temporal information in Linked Data cloud, we analyse the latest release of the BTC dataset which was crawled from the Web in May/June 2011 using a random sample of URIs from the BTC 2010 dataset as seed URIs. The BTC corpus which represents only a part of all available Linked Data on the Web, contains over 2.1 bn statements in N-Quads[5] format with over 47 K unique predicates, collected from 7.4 M RDF documents. However, our corpus constitutes a large collection of documents sampled from a wide variety of Linked Data publishers. A crawling-based approach is per design biased towards datasets that are well-interlinked, while more isolated datasets are less likely to be found. We also observe that the corpus is static, and it samples only RDF/XML, not covering data in other syntaxes like RDFa. We expect these aspects not to have any negative effects on the findings of our analysis, which still targets specifically prominent and well interlinked part of the LOD cloud.

Considering the size of the corpus, we use Apache Hadoop[6] to analyse the data. Hadoop allows for the parallel and distributed processing of large datasets across clusters of computers. We run the analysis on the KIT OpenCirrus[7] Hadoop cluster. For our analysis we used 54 work nodes, each with a 2.27 GHz 4-Core CPU and 100GB RAM, a setup which completes a scan over the entire corpus in about 15 minutes.

3.2 General Analysis

To gather a broad selection of temporal information in BTC, we employ a string-based search method which implements a class named SimpleDateFormat[8] in Java. We are confident about the correctness of the collected data because the time parser is well-known and used by a large community.

We assume that if temporal information is present, it is contained in the object position of quads. Thus, we use regular expressions to identify temporal information in the object of every quad in the BTC. However, it has been recently shown that the best practices used to publish data on the Web [3] are not always followed by publishers [13].

We notice that often RDF publishers do not use the date formats defined by standards such as RFC 822, ISO 8601 or XML Schema. In order to collect all temporal information that is represented in the BTC but is not fully compliant to standard date formats, we consider variations of the standards. The variations of the standard date formats are expressed by regular expressions based on the following patterns: `(EEE), dd MMM yy (HH:mm:(ss) (Z|z))` and `yyyy-MM-(dd('T'HH:mm:(ss).(s)(Z|z)))` respectively[9]. We extract

[5] `http://sw.deri.org/2008/07/n-quads/`
[6] `http://hadoop.apache.org/`
[7] `https://opencirrus.org/`
[8] `http://docs.oracle.com/javase/6/docs/api/java/text/SimpleDateFormat.html`
[9] The value in the parentheses is optional.

Table 1. Top twenty PLDs with respect to temporal quads

PLD	quad. (M)	Tquad (K)	doc (K)	Tdoc (K)
scinets.org	56.2	3,391	51.9	44.3
legislation.gov.uk	33.1	1,249	246.4	246.4
ontologycentral.com	55.3	1,029	4.6	4.4
bibsonomy.org	34.5	881	234.7	177.3
loc.gov	7.8	854	345.3	302.9
bbc.co.uk	6.3	679	173.5	83.6
livejournal.com	169.8	530	239.2	238.9
rdfize.com	37.6	495	204.7	204.6
data.gov.uk	13.8	479	178.8	91.9
dbpedia.org	28.4	423	596.6	124.1
musicbrainz.org	2.5	359	0.3	0.3
tfri.gov.tw	153.3	272	154.4	78.2
archiplanet.org	16.3	186	79.2	53.5
freebase.com	27.8	173	572.9	109.1
vu.nl	6.8	156	294.2	26.7
fu-berlin.de	5.7	139	291.6	37.4
bio2rdf.org	20.2	129	744.7	71.6
blogspace.com	0.9	124	0.2	0.2
opera.com	24.1	124	160.3	124.1
myexperiment.org	1.5	114	26.1	13.7

Table 2. Top twenty temporal properties wrt. temporal quads

Temporal Property	quad (M)	doc (K)
dcterms:#modified	3.4	44
dcterms:modified	2.3	842
dcterms:date	1.5	247
dc:date	1.4	188
dcterms:created	0.6	450
dcterms:issued	0.2	222
lj:dateCreated	0.2	238
swivt:#creationDate	0.2	197
lj:dateLastUpdated	0.22	225
wiki:Attribute3ANRHP _certification_date	0.18	53
tl:timeline.owl#start	0.17	31
tl:timeline.owl#end	0.15	24
bio:date	0.14	143
po:schedule_date	0.14	15
swrc:ontology#value	0.096	37
cordis:endDate	0.078	0.002
nl:currentLocationDateStart	0.076	26
po:start_of_media_availability	0.074	10
foaf:dateOfBirth	0.068	68
liteco:dateTime	0.062	62

12,863,547 *temporal quads*, i.e., quads containing a temporal entity, and 1,670 unique temporal properties from the corpus.

Furthermore, to provide a deeper analysis of the distribution of temporal information within the dataset, we extract all the pay-level domains (PLDs) occurring in the context of the quads. Herein, we use PLDs to distinguish individual data providers [20]. Table 1 lists the top 20 PLDs publishing the largest number of temporal quads. For each PLD we report: the total number of quads (*quad.* in Table 1), the number of temporal quads (*Tquad.*), the number of documents (*doc*) and the number of temporal documents (*Tdoc*).

We can notice that although `scinets.org` is listed on top of the list, it does not provide the highest ratio of temporal quads over the total number of quads compared to other datasets. With respect to the temporal quads, we can notice that `musicbrainz.org` and `blogspace.com` represent the largest number of temporal quads as a proportion of all quads. Similarly for the documents, we notice that `legislation.gov.uk`, `rdfize.com` and `blogspace.com` represent the three PLDs with the largest number of temporal documents as a proportion of all documents.

Table 2 lists the top 20 temporal properties that occur more frequently in the BTC, reporting the number of quads and documents they occur in. We also provide an analysis of the distribution of the top-10 most frequent temporal properties within the most significant PLDs, which is plotted in Figure 1. It can

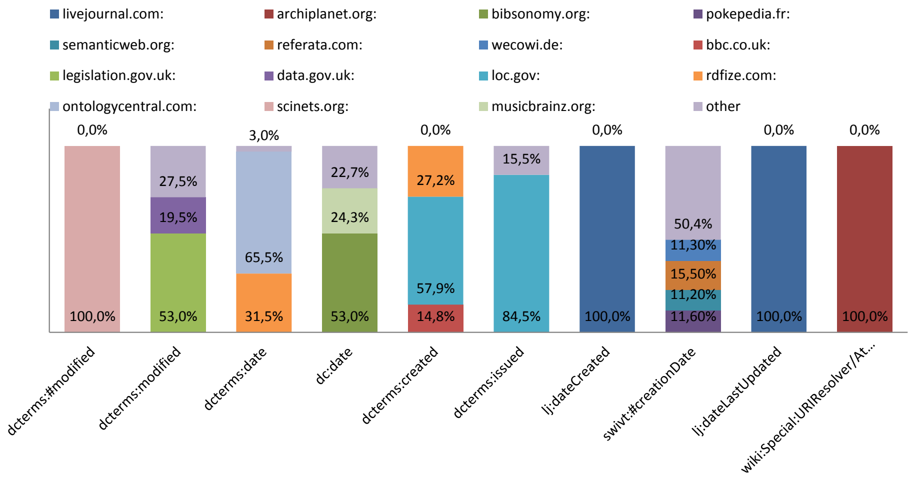


Fig. 1. Distribution of top ten temporal properties with respect to main PLDs

be noticed that not only the properties of the Dublin Core (DC) vocabulary[10] do occur much more frequently than other properties, but they are also used more often across different datasets. Remarkably, the temporal property that occurs more often in the BTC dataset, i.e., `dcterms:#modified`, has a wrong spelling (the correct spelling denotes in fact the second most frequent temporal property in the corpus). As shown in Figure 1, this is also the only temporal property published in the `scinets.org` context, and the spelling is wrong in all the quads having the same context.

4 Temporal Meta-information Description Models

In this section we focus on temporal meta-information, that is temporal information defined as T(x,a,t) where x can be either a statement or a graph. Because of the tight constraints given by the triple-based structure of RDF descriptions, the concrete RDF-based representation of an even simple temporal annotation like $T(x, a, t)$, with x being a document and t a temporal entity, requires some sophisticated mechanisms. Several approaches for providing a concrete representation of a temporal annotation have been proposed. We identify three core perspectives that have been adopted for the concrete representation of temporal meta-information in RDF:

– Document-centric Perspective, where time points are associated with RDF documents.
– Fact-centric Perspective, where time points or intervals (usually intervals) are associated with facts; since facts can be represented by one or more statements - we further separate the Fact-centric Perspective into:

[10] `http://www.dublincore.org/documents/dces/`

- Sentence-centric Perspective, which explicitly define the temporal validity of one or more statements annotating them with time points or intervals.
- Relationship-centric Perspective, which encapsulates time points or intervals into objects representing n-ary relations.

In the following we explain in detail the approaches proposed according to the aforementioned perspectives.

4.1 Document-Centric Perspective

Graphs, i.e. RDF documents, can be associated with temporal meta-information following two approaches: the first one uses HTTP-metadata, and in particular the Last-modified field of the HTTP response header; the second one expresses temporal meta-information using RDF statements with temporal properties taken from available vocabularies such as Dublin Core. Temporal meta-information following these approaches, and in particular, *Last-modified* and *ETage* properties of HTTP headers have been used for the detection of changes in Web documents publishing RDF data [29].

Protocol-based representation. A Protocol-based representation adopts point-based time modelling; the temporal meta-information is not persistently associated with a Web document, but can be extracted from the HTTP header returned in response to an HTTP GET request for the document. The temporal meta-information associates a time point, represented by a date, with a Web document G using a predicate a defined in the HTTP protocol according to the schema defined as follows:

```
HTTP Response Header
Status: HTTP/1.1 200 OK
```
a : t^p

Metadata-based representation. Let $\langle s, p, o\rangle$ be a statement, u_G a named graph, a_G a temporal property, t^p a time point; the Metadata-based representation associates a temporal meta-information with an RDF document as follows:

$$\langle s, p, o, u_G\rangle$$
$$\langle u_G, a_G, t^p, u_G\rangle$$

Examples of datasets providing temporal meta-information to the documents are: Protein knowledge base (UNIPROT) and legislation.gov.uk.

4.2 Fact-Centric Perspective

In the Fact-centric Perspective facts are associated with temporal meta-information that constrain their temporal validity. The first RDF model proposed to formally capture this idea is Temporal RDF [11]. In this model, RDF statements are annotated with time intervals constraining their temporal validity; the intervals are interpreted over a point-based, discrete and linearly ordered temporal domain.

Temporal RDF-based representation. Let $\langle s, p, o\rangle$ be an RDF statement and $[t^b; t^e]$ a time interval with a starting point t^b and an ending point t^e, a Temporal RDF-based representation is a temporal annotated statement having the form $\langle s, p, o\rangle[t^b{:}t^e]$.

The encoding of the above definition into the triple-based RDF data model is not straightforward because RDF can "natively" represents only binary relations. In order to solve this problem, several approaches for encoding the temporal validity of facts into the standard RDF syntax have been proposed. These approaches follow two perspectives that present significant differences: the Sentence-centric Perspective and the Relationship-centric Perspective.

Sentence-Centric Perspective

Two strategies are adopted to represent the temporal validity of fact adopting the Sentence-centric Perspective.

Reification-based representation. Let $\langle s, p, o\rangle$ be a statement, s^{st} an identifier of a statement, a^b_S and a^e_S two temporal properties, and $[t^b{:}t^e]$ a time interval; a Reification-based representation is defined as follows:

$\langle s^{st}, \texttt{rdf:type}, \texttt{rdf:Statement}\rangle$
$\langle s^{st}, \texttt{rdf:subject}, s\rangle$
$\langle s^{st}, \texttt{rdf:predicate}, p\rangle$
$\langle s^{st}, \texttt{rdf:object}, o\rangle$
$\langle s^{st}, a^b_S, t^b\rangle$
$\langle s^{st}, a^e_S, t^e\rangle$

The first four sentences encode the reification of the statement representing the fact using the RDF vocabulary. The temporal properties a^b_S and a^e_S link the statements respectively to the beginning and the ending point of the time interval $[t^b{:}t^e]$ associated with the statement. Notice that a property a_S can have a time point or a time interval as property value. As an example of datasets adopting such approach we mention Timely Yago [30].

In the above approach, every sentence associated with a temporal annotation has to be reified. An alternative approach allows grouping together statements that have the same temporal validity by introducing the concept of *temporal graph* [28]. Temporal graphs are named graphs annotated with timeintervals; each time interval is represented by exactly one temporal graph, where all triples belonging to this graph share the same validity period. Temporal meta-information are collected in a *default graph* which occur as context in the quads.

Applied Temporal RDF-based representation. Let u_{TG} and u_G be the names respectively of a temporal graph and of the default graph, a^b_S and a^e_S two temporal properties, $[t^b{:}t^e]$ a time interval and $\langle s, p, o\rangle$ a statement; the Applied temporal RDF-based representation is defined as follows:

$\langle u_{TG}, a^b_S, t^b, u_G \rangle$
$\langle u_{TG}, a^e_S, t^e, u_G \rangle$
$\langle s, p, o, u_{TG} \rangle$

The temporal properties a^b_S and a^e_S link the temporal graph respectively to the beginning and the ending point of the time interval $[t^b{:}t^e]$. More statements can be associated with the same temporal graph. As an example of dataset that uses such approach is EvOnt [28].

Relationship-Centric Perspective

N-ary Relationship design patterns[11] are introduced to represent RDF relations with arity greater than two. These patterns model an n-ary relation with a set of RDF statements by (i) introducing a specific resource to identify the relation, and (ii) creating links between this resource and the constituents of the relation (resources and literals). These patterns can be used to associate temporal annotations with facts represented by RDF statements to constrain their temporal validity. For example, the fact "Alessandro Del Piero (ADP) plays for Juventus", which is valid within the time interval [1993,2012], can be modelled as a quintuple ⟨ADP, playsFor,Juventus,1993,2012⟩ and represented following the N-ary Relationship pattern. A resource r is introduced to identify the relation and the temporally annotated fact can be represented by the set of RDF statements ⟨ADP,playsFor,r⟩, ⟨r,team,Juventus⟩, ⟨r,from,1993⟩, ⟨r,to,2012⟩. The direction of the links and the strategies adopted for naming the properties can change according to different variants of the pattern [19,25]. However, the temporal annotations are linked to the resources that identify a relation in all the proposed variants. In this paper we define the N-ary Relationship-based representation adopting the variant described in the second use case of the W3C document, the one that occurs more frequently in the BTC corpus.

N-ary-relationship-based representation. Let $\langle s, p, o \rangle$ be an RDF statement, r a new resource, p_1 and p_2 two properties, a^b_R and a^e_R two temporal properties, and $[t^b{:}t^e]$ a time interval; the N-ary-relationship-based representation is defined as follows:

$\langle s, p_1, r \rangle$
$\langle r, p_2, o \rangle$
$\langle r, a^b_R, t^b \rangle$
$\langle r, a^e_R, t^e \rangle$

Although p_1 and p_2 can be two new properties, one of the two is usually equal to p as in the example discussed above. As an example of dataset we mention Freebase[12].

A second approach to model temporal meta-information according to the Fact-centric perspective is based on the concepts of *fluent* and *timeslice* [31]. Fluents

[11] http://www.w3.org/TR/swbp-n-aryRelations/
[12] http://www.freebase.com/

are properties that hold at a specific moment in time, i.e., object properties that change over time. The properties representing fluents link two timeslices, i.e., entities that are extended through temporal dimensions.

4D-fluents-based representation. Let $\langle s, p, o\rangle$ be an RDF statement, a_R^b and a_R^e two temporal properties, $[t^b{:}t^e]$ a time interval, and s^t and o^t two timeslices associated respectively with s and o; the 4D-fluents-based representation is defined as follows:

$\langle s^t$, `rdf:type`, `:TimeSlice`$\rangle$
$\langle s$, `:hasTimeslice`, $s^t\rangle$
$\langle s^t, a_R^b, t^b\rangle$
$\langle s^t, a_R^e, t^e\rangle$
$\langle o^t$, `rdf:type`, `:TimeSlice`$\rangle$
$\langle o$, `:hasTimeslice`, $o^t\rangle$
$\langle o^t, a_R^b, t^b\rangle$
$\langle o^t, a_R^e, t^e\rangle$
$\langle s^t, p, o^t\rangle$

Although we could not find any dataset adopting this approach, well-known ontologies like PROTON[13] and DOLCE[14] adopt it.

5 Quantitative and Qualitative Analysis

In this section we analyse and evaluate the adoption of the approaches for representing temporal meta-information. Our quantitative analysis is augmented by a qualitative discussion in Section 5.3, based on both experiments and literature, to highlight the advantages and disadvantages of each approach.

Please observe that some approaches cannot be detected automatically in the data. Therefore, for certain constructs we select a random sample and manually identify the constructs in the sample. We then scale the resulting measure to the entire dataset, which consists of 2.1bn quads in 7.4M documents. Of those, 12.8M were temporal quads (containing a date literal) occurring in 2.5M documents.

Analysing larger samples is infeasible due to the high manual effort involved in checking for constructs in the entire dataset; please note that random sampling is an established method for estimating properties of large populations (e.g., the prediction of election outcomes use small samples and achieve sufficient accuracy [2]). For instance, the error bound for Protocol-based representation is +/- 1.9%. The samples used in the experiments are available online[15].

Not all surveyed approaches are adopted on the web. We did not find any uses of the Applied temporal RDF-based representation and the 4D-fluents-based representation in the data. Table 3 gives an overview of our findings.

[13] `http://proton.semanticweb.org/`
[14] `http://www.loa.istc.cnr.it/DOLCE.html`
[15] `http://people.aifb.kit.edu/sts/data/`

Table 3. Temporal meta-information representation approaches and the respective occurrence compared to i) quads having temporal information; ii) overall quads in the BTC; iii) overall documents in the BTC (n/a = not applicable, - = no occurrence).

Perspective	Approach	Occurrence temp. quads	Occurrence overall quads	Occurrence overall docs
Document	Protocol	n/a	n/a	9.5%
	Metadata	5.1%	0.00019%	0.56%
Fact	Reification	0.02%	0.0000008%	0.006%
	Applied temporal RDF	-	-	-
	N-ary relationship	12.24%	0.0005%	0.6%
	4D-fluents	-	-	-

5.1 Document-Centric Perspective

To identify the use of the *Protocol-based representation* we ascertain how many of the URIs that identified documents in the BTC return date information in the HTTP header. We generate a random sample of 1000 documents (from the context of the quads), and for each document URI in the sample we perform an HTTP lookup to check the last-modified header in the HTTP response. We found that only 95 out of 1000 URIs returned last-modified headers.

To identify the use of the *Metadata-based representation*, we select a sample of 1000 URIs that appear in the subject position of quads with temporal information. We need to ensure that those subject URIs are in fact documents (information resources), as the Metadata-based representation pattern is concerned with documents. Thus, from the sample we exclude URIs containing the # symbol (as URIs with a # per definition do not refer to a document).

For the remaining URIs we send an HTTP request and analyse the response code to determine whether the URI identified a document. We found that 432 (43.2%) identified documents (i.e., directly returned a 200 OK status code). These information resources are not limited to RDF but they also include resources in other formats such as HTML, MP3, XML or PDF. We manually check for RDF documents with only the temporal meta-information such as modified and updated, which resulted in 51 documents.

Of the 51 RDF documents with temporal meta-information in HTTP headers, 43 are also associated with metadata-based dates. Thus, for each of the 43 identified documents we compared protocol-based last-modified and metadata-based last-modified dates. We found that protocol-based last-modified dates are more up-to-date compared to metadata-based dates with an average of almost a year (364 days).

5.2 Fact-Centric Perspective

We analyse the *Reification-based representation* in the BTC by looking for how often reified statements contain temporal information. The pattern first identifies the quads containing predicates that are defined in the RDF reification vocabulary (i.e., `rdf:subject`, `rdf:predicate`, and `rdf:object`). From the identified cases we extract only those reified statement that have temporal meta-information associated with their subjects. In the entire BTC dataset we found 2,637 reified statements containing temporal meta-information.

To account for *N-ary-relationship-based representation* we again use a combination of sampling of the results of a query over the dataset with manual verification since n-ary relations are impossible to identify just by analysing the graph structure. Hence, we sample and manually identify occurrences.

The following pattern identifies for each document triples of the form $\langle s, p, o \rangle$ and $\langle o, p^*, o^* \rangle$ and furthermore identifies whether o is also associated with a temporal entity. Notice that the possibility to join two triples x and y where $x.object = y.subject$ is a necessary, but not sufficient condition, to identify n-ary relations. All results are contained in a set that we name *scoped set* consisting of 7M temporal quads. Hence, from the scoped set, we select three different random samples of 100 triples and we manually verify if respective documents identify an n-ary relation. Results of such manual analysis show that 10, 10 and 12 out of 100 triples in the samples are used with an n-ary relation.

5.3 Discussion and Recommendations

In the following we discuss the results and provide recommendations for data publishers and consumers.

The approaches that are part of the Document-centric Perspective are more extensively adopted than the approaches of the Fact-centric Perspective. As we hypothesised, the number of temporal meta-information associated with documents is greater than those associated with facts. Still, the use of temporal meta-information for documents (about 10% overall) are not sufficiently high enough to support our outlined use case.

We identify two approaches used for annotating documents with temporal meta-information: the ***Protocol-based representation*** and the ***Metadata-based representation***. We notice that the number of temporal meta-information are much more available in the Protocol-based rather than the Metadata-based representation. The temporal meta-information in the HTTP header, when available, are more up-to-date than the ones in the RDF document itself. *Consumers:* The applications that consume temporal meta-information should first check for temporal meta-information in the Protocol-based representation because they are more up-to-date; in case this information is not available the applications should be able to check in the Metadata-based representation. *Publishers:* Publishers should carefully update the temporal meta-information whenever the data in the document is changed; temporal meta-information in both Protocol- and Metadata-based representation should be consistent.

We identify four approaches used for annotating facts with temporal meta-information, grouped into the Sentence-centric Perspective and the Relationship-centric Perspective. These approaches associate validity expressed as temporal entities to facts.

The use of the ***Reification-based representation*** show a high complexity w.r.t. query processing [14]. The approach appears only in a very small number of quads. *Consumers:* Consumers should be able to evaluate based on the application scenario (e.g., the expected types of queries) if it is possible to either build their applications over such representation or to choose a different, and more efficient approach (e.g. Applied temporal RDF-based representation). *Publishers:* Publishers should be aware that best practices discourage the use of Reification-based representations, as they are cumbersome to use in SPARQL queries [3], even though they may be useful for representing temporal meta-information.

The performance of ***Applied temporal RDF-based representation*** has been reported to have still some efficiency issues [28], especially in the worst case, when the number of graphs (which are associated with temporal annotations) is almost equivalent to the number of triples. *Consumers:* Although we found no usage of the Applied temporal RDF-based representation in the BTC, the approach should deserve more attention because it supports expressive temporal queries based on τ-SPARQL, and can be applied to datasets that provide temporal information according to a Reification-based representation. *Publishers:* Publishers should take into consideration the worst case when using the Applied temporal RDF-based representation. Therefore, they should use it only when it is possible to group a considerable number of triples into a single graph.

The ***N-ary-relationship-based representation*** embeds time in an object that represents a relation. In the BTC, 0.6% of documents contain at least one case of N-ary-relationship-based representation, which is greater than the Reification-based representation but still represents only a small fraction of the overall number of documents. *Consumers:* Consumer applications can evaluate the temporal validity of facts from representations based on this approach. The lack of a clear distinction between plain temporal information and temporal meta-information provides high flexibility, but at the same makes difficult to predict the kind of temporal information that can be leveraged and interpret its meaning. Collecting these temporal meta-information with automatic methods is not straightforward, as shown by the manual efforts required in our analysis to identify this information. *Publishers:* Many situations require temporal meta-information associated with relations that can be modelled only as complex objects. Therefore, we recommend to publishers to use N-ary-relationship-based representation for complex modelling tasks because it allows flexibility on representing temporal meta-information associated with relation.

The ***4D-fluents-based representation*** supports advanced reasoning functionalities, but, probably also because of its complexity, has not been adopted on the Web.

6 Conclusion

The key contribution of this paper is the investigation of temporal information in Linked Data on the Web, which is important for several research and application domains. As time introduces a further dimension to the data it cannot be easily represented in RDF, a language based on binary relations; as a result, several approaches for representing temporal information have been proposed. Based on the qualitative and quantitative analysis using the Billion Triple Challenge 2011 dataset, we came to the conclusion that the availability of temporal information describing the history and the temporal validity of statements and graphs is still very limited. If the representation of temporal validity of RDF data is somewhat more complex and can be expected to be considered in specific contexts, information about the creation and modification of data can be published with quite simple mechanisms. Yet, this information would have great value, e.g., when data coming from different sources need to be integrated and fused.

As future work, we plan to develop automatic techniques for the assessment of temporal data qualities in Linked Data, such as data currency and timeliness. With the deeper understanding of temporal information gained through our present analysis, we aim to capture and process a large amount of temporal information, overcoming several limitations of preliminary work [26].

Acknowledgements. We thank Basil Ell, Julia Hoxha and Sebastian Rudolph for their valuable comments and acknowledge the support of the EC's Seventh Framework Programme FP7/2007-2013 (PlanetData, Grant 257641).

References

1. Alonso, O., Strötgen, J., Baeza-Yates, R., Gertz, M.: Temporal Information Retrieval: Challenges and Opportunities. In: 1st Temporal Web Analytics Workshop at WWW, pp. 1–8 (2011)
2. Bartlett, J., Kotrlik, I., Higgins, C.: Organizational Research: Determining Appropriate Sample Size in Survey Research. Information Technology, Learning, and Performance Journal, 43 (2001)
3. Bizer, C., Cyganiak, R., Heath, T.: How to publish Linked Data on the Web. linkeddata.org Tutorial (2008)
4. Cho, J., Garcia-Molina, H.: The Evolution of the Web and Implications for an Incremental Crawler. In: The 26th VLDB, pp. 200–209 (2000)
5. Correndo, G., Salvadores, M., Millard, I., Shadbolt, N.: Linked Timelines: Temporal Representation and Management in Linked Data. In: 1st International Workshop on Consuming Linked Data at ISWC (2010)
6. Crocker, D.H.: Standard for the Format of ARPA Internet Text Messages. RFC 822 (1982)
7. De Sompel, H.V., Sanderson, R., Nelson, M.L., Balakireva, L., Shankar, H., Ainsworth, S.: An HTTP-Based Versioning Mechanism for Linked Data. In: 3rd Linked Data on the Web Workshop at WWW (2010)
8. Fallside, D.C., Walmsley, P.: XML Schema Part 0: Primer Edition, 2nd edn. World Wide Web Consortium (2004)
9. Grandi, F.: Introducing an annotated bibliography on temporal and evolution aspects in the World Wide Web. SIGMOD Record, 84–86 (2004)
10. Gutierrez, C., Hurtado, C., Mendelzon, A.O., Pérez, J.: Foundations of Semantic Web Databases, pp. 520–541 (2011)
11. Gutierrez, C., Hurtado, C.A., Vaisman, A.A.: Temporal RDF. In: Gómez-Pérez, A., Euzenat, J. (eds.) ESWC 2005. LNCS, vol. 3532, pp. 93–107. Springer, Heidelberg (2005)

12. Hobbs, J., Pan, F.: An Ontology of Time for the Semantic Web. Processings of the ACM Transactions on Asian Language Information, 66–85 (2004)
13. Hogan, A., Harth, A., Passant, A., Decker, S., Polleres, A.: Weaving the Pedantic Web. In: 3rd Linked Data on the Web Workshop at WWW (2010)
14. Hogan, A., Umbrich, J., Harth, A., Cyganiak, R., Polleres, A., Decker, S.: An Empirical Survey of Linked Data Conformance. Web Semantics (2012)
15. Horton, M.R.: Standard for Interchange of USENET Messages. RFC 850, Internet Engineering Task Force (1983)
16. ISO 8601. Data Elements and Interchange Formats-Information Interchange-Representation of Dates and Times (2004)
17. Käfer, T., Umbrich, J., Hogan, A., Polleres, A.: Towards a Dynamic Linked Data Observatory. In: 5th Linked Data on the Web Workshop at WWW (2012)
18. Kline, N.: An Update of the Temporal Database Bibliography. SIGMOD Record, 66–80 (1993)
19. Koubarakis, M., Kyzirakos, K.: Modeling and Querying Metadata in the Semantic Sensor Web: The Model stRDF and the Query Language stSPARQL. 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. 425–439. Springer, Heidelberg (2010)
20. Lee, H.T., Leonard, D., Wang, X., Loguinov, D.: IRLbot: Scaling to 6 Billion Pages and Beyond. In: The 17th WWW, pp. 427–436 (2008)
21. Li, P., Dong, X.L., Maurino, A., Srivastava, D.: Linking temporal records. The VLDB Endowment (2011)
22. Mendes, P.N., Mühleisen, H., Bizer, C.: Sieve: Linked Data Quality Assessment and Fusion. In: 2nd International Workshop on Linked Web Data Management at EDBT (2012)
23. Panziera, L., Comerio, M., Palmonari, M., De Paoli, F., Batini, C.: Quality-Driven Extraction, Fusion and Matchmaking of Semantic Web API Descriptions. J. Web Eng. 11(3), 247–268 (2012)
24. Popitsch, N., Haslhofer, B.: DSNotify - A Solution for Event Detection and Link Maintenance in Dynamic Datasets. Web Semantics, 266–283 (2011)
25. Rodrıguez, A., McGrath, R., Liu, Y., Myers, J.: Semantic Management of Streaming Data. In: 2nd International Workshop on Semantic Sensor Networks at ISWC (2009)
26. Rula, A., Palmonari, M., Maurino, A.: Capturing the Age of Linked Open Data: Towards a Dataset-independent Framework. In: 1st International Workshop on Data Quality Management and Semantic Technologies at IEEE ICSC (2012)
27. Sheth, A., Henson, C., Sahoo, S.: Semantic Sensor Web. IEEE Internet Computing 12(4), 78–83 (2008)
28. Tappolet, J., Bernstein, A.: Applied Temporal RDF: Efficient Temporal Querying of RDF Data with SPARQL. 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. 308–322. Springer, Heidelberg (2009)
29. Umbrich, J., Hausenblas, M., Hogan, A., Polleres, A., Decker, S.: Towards Dataset Dynamics: Change Frequency of Linked Open Data Sources. In: 3rd Linked Data on the Web Workshop at WWW (2010)
30. Wang, Y., Zhu, M., Qu, L., Spaniol, M., Weikum, G.: Timely YAGO: Harvesting, Querying, and Visualizing Temporal Knowledge from Wikipedia. In: The 13th EDBT, pp. 697–700 (2010)
31. Welty, C., Fikes, R., Makarios, S.: A Reusable Ontology for Fluents in OWL. In: Frontiers in Artificial Intelligence and Applications, p. 226 (2006)

Semantic Sentiment Analysis of Twitter

Hassan Saif, Yulan He, and Harith Alani

Knowledge Media Institute, The Open University, United Kingdom
{h.saif,y.he,h.alani}@open.ac.uk

Abstract. Sentiment analysis over Twitter offer organisations a fast and effective way to monitor the publics' feelings towards their brand, business, directors, etc. A wide range of features and methods for training sentiment classifiers for Twitter datasets have been researched in recent years with varying results. In this paper, we introduce a novel approach of adding semantics as additional features into the training set for sentiment analysis. For each extracted entity (e.g. iPhone) from tweets, we add its semantic concept (e.g. "Apple product") as an additional feature, and measure the correlation of the representative concept with negative/positive sentiment. We apply this approach to predict sentiment for three different Twitter datasets. Our results show an average increase of F harmonic accuracy score for identifying both negative and positive sentiment of around 6.5% and 4.8% over the baselines of unigrams and part-of-speech features respectively. We also compare against an approach based on sentiment-bearing topic analysis, and find that semantic features produce better Recall and F score when classifying negative sentiment, and better Precision with lower Recall and F score in positive sentiment classification.

Keywords: Sentiment analysis, semantic concepts, feature interpolation.

1 Introduction

The emergence of social media has given web users a venue for expressing and sharing their thoughts and opinions on all kinds of topics and events. Twitter, with nearly 600 million users[1] and over 250 million messages per day,[2] has quickly become a gold mine for organisations to monitor their reputation and brands by extracting and analysing the sentiment of the Tweets posted by the public about them, their markets, and competitors.

Sentiment analysis over Twitter data and other similar microblogs faces several new challenges due to the typical short length and irregular structure of such content. Two main research directions can be identified in the literature of sentiment analysis on microblogs. First direction is concerned with finding new methods to run such analysis, such as performing sentiment label propagation on Twitter follower graphs [14], and employing social relations for user-level sentiment analysis [15,5]. The second direction is focused on identifying new sets of features to add to the trained model for sentiment identification, such as microblogging features including hashtags, emoticons [2], the presence of intensifiers such as all-caps and character repetitions [6] etc., and sentiment-topic features [12].

[1] twopcharts.com/twitter500million.php
[2] www.geekosystem.com/twitter-250-million-tweets-per-day

The work in this paper falls into the second direction, by investigating a novel set of features derived from the semantic conceptual representation of the entities that appear in tweets. The *semantic features* consist of the semantic concepts (e.g. “person”, “company”, “city”) that represent the entities (e.g. “Steve Jobs”, “Vodafone”, “London”) extracted from tweets. The rational behind introducing these features is that certain entities and concepts tend to have a more consistent correlation with positive or negative sentiment. Knowing these correlations can help determining the sentiment of semantically relevant or similar entities, and thus increasing accuracy of sentiment analysis. To the best of our knowledge, using these semantic features in the model training for sentiment analysis has not been explored before. We evaluated three popular tools for entity extraction and concept identification; AlchemyAPI,[3] Zemanta,[4] and OpenCalais,[5] and used the one that performed best in terms of quantity and accuracy of the identified concepts.

While previous work on feature engineering for sentiment classification on tweets [1,6] simply incorporate features through augmentation, our experimental results show that it is more effective to incorporate semantic features through interpolation. Hence we incorporate the semantic features into Naïve Bayes (NB) model training using an interpolation approach.

We experiment and evaluate our proposed approach with three datasets collected from Twitter; a general Stanford Twitter Sentiment (STS) dataset, a dataset on the Obama-McCain Debate (OMD), and one on Health Care Reform (HCR). Our results show that combining our semantic features with word unigrams outperforms the baseline model trained from unigrams only across all three datasets by an average accuracy of 6.47%. It also outperforms the accuracy of sentiment analysis using the common part-of-speech (POS) features often used in the literature [9,1] by an average of 4.78%. Although these improvements may appear modest, they are very notable in comparison to the scale of improvements reported in similar literatures. Our results show that the advantage of using semantic features in microblog sentiment analysis over other techniques is mostly restricted to negative sentiment identification, in large topically-diverse datasets.

The main contributions of this paper can be summarised as follows:

– Introduce and implement a new set of semantic features for training a model for sentiment analysis of tweets.
– Investigate three approaches for adding such features into the training model; by replacement, by argumentation, and by interpolation, and show the superiority of the latter approach.
– Test accuracy of sentiment identification when using semantic features with unigrams on three Twitter datasets, and produce an average harmonic mean (F score) accuracy of 75.95%, with 77.18% Precision and 75.33% Recall.
– Demonstrate the value of *not* removing stowords in increasing sentiment identification accuracy.
– Show an average of 6.47% increase in the F score against a baseline approach based on unigrams only.

[3] www.alchemyapi.com
[4] www.zemanta.com
[5] www.opencalais.com

- Show an average of 4.78% increase in F score in comparison to using the common POS features alongside unigrams.
- Compare results with sentiment-bearing topic features [12] and show that semantic features improve F by 1.22% when identifying negative sentiment, but worsens F by 2.21% when identifying positive sentiment.

The rest of the paper is organised as follows. Section 2 outlines existing work on sentiment analysis with focus on twitter sentiment analysis. Section 3 describes the three Twitter datasets used in our experiments. Section 4 presents our proposed approach of using semantic features for sentiment analysis, and describes three methods for incorporating these features into the sentiment classifier. In Section 5 we describe the baselines we use for evaluating and comparing our results. Experimental results are fully detailed and discussed in Section 6. Discussion and future work are covered in Section 7. Finally, we conclude our work in Section 8.

2 Related Work

Sentiment analysis of tweets data is considered as a much harder problem than that of conventional text such as review documents. This is partly due to the short length of tweets, the frequent use of informal and irregular words, and the rapid evolution of language in Twitter. A large amount of work has been conducted in Twitter sentiment analysis following the feature-based approaches. Go et al. [4] explored augmenting different n-gram features in conjunction with POS tags into the training of supervised classifiers including Naive Bayes (NB), Maximum Entropy (MaxEnt) and Support Vector Machines (SVMs). They found that MaxEnt trained from a combination of unigrams and bigrams outperforms other models trained from a combination of POS tags and unigrams by almost 3%. However, a contrary finding was reported in [9] that adding POS tag features into n-grams improves the sentiment classification accuracy on tweets.

Barbosa and Feng [2] argued that using n-grams on tweet data may hinder the classification performance because of the large number of infrequent words in Twitter. Instead, they proposed using microblogging features such as re-tweets, hashtags, replies, punctuations, and emoticons. They found that using these features to train the SVMs enhances the sentiment classification accuracy by 2.2% compared to SVMs trained from unigrams only. A similar finding was reported by Kouloumpis et al. [6]. They explored the microblogging features including emoticons, abbreviations and the presence of intensifiers such as all-caps and character repetitions for Twitter sentiment classification. Their results show that the best performance comes from using the n-grams together with the microblogging features and the lexicon features where words tagged with their prior polarity. However, including the POS features produced a drop in performance.

Agarwal et al. [1] also explored the POS features, the lexicon features and the microblogging features. Apart from simply combining various features, they also designed a tree representation of tweets to combine many categories of features in one succinct representation. A partial tree kernel [8] was used to calculate the similarity between two trees. They found that the most important features are those that combine prior polarity of words with their POS tags. All other features only play a marginal role. Furthermore, they also showed that combining unigrams with the best set of features outperforms the tree kernel-based model and gives about 4% absolute gain over a unigram baseline.

Rather than directly incorporating the microblogging features into sentiment classifier training, Speriosu et al. [14] constructed a graph that has some of the microblogging features such as hashtags and emoticons together with users, tweets, word unigrams and bigrams as its nodes which are connected based on the link existence among them (e.g., users are connected to tweets they created; tweets are connected to word unigrams that they contain etc.). They then applied a label propagation method where sentiment labels were propagated from a small set of nodes seeded with some initial label information throughout the graph. They claimed that their label propagation method outperforms MaxEnt trained from noisy labels and obtained an accuracy of 84.7% on the subset of the Twitter sentiment test set from [4].

Existing work mainly concentrates on the use of three types of features; lexicon features, POS features, and microblogging features for sentiment analysis. Mixed findings have been reported. Some [9,1] argued the importance of POS tags with or without word prior polarity involved, while others emphasised the use of microblogging features [2,6]. In this paper, we propose a new type of features for sentiment analysis, called semantic features, where for each entity in a tweet (e.g. iPhone, iPad, MacBook), the abstract concept that represents it will be added as a new feature (e.g. Apple product). We compare the accuracy of sentiment analysis against other types of features; unigrams, POS features, and the sentiment-topic features. To the best of our knowledge, using such semantic features is novel in the context of sentiment analysis.

3 Datasets

For the work and experiments described in this paper, we used three different Twitter datasets as detailed below. The statistics of the datasets are shown in Table 1.

Table 1. Statistics of the three Twitter datasets used in this paper

Dataset	Type	No. of Tweets	Positive	Negative
Stanford Twitter Sentiment Corpus (STS)	Train	60K	30K	30K
	Test	1,000	470	530
Health Care Reform (HCR)	Train	839	234	421
	Test	839	163	536
Obama-McCain Debate (OMD)	n-fold cross validation	1,081	393	688

Stanford Twitter Sentiment Corpus (STS)

This dataset consists of 60,000 tweets randomly selected from the Stanford Twitter Sentiment corpus (STS) [4]. Half of the tweets in this dataset contains positive emoticons, such as :), :-), :), :D, and =), and the other half contains negative emoticons such as :(, :-(, or : (. The original dataset from [4] contained 1.6 million general tweets, and its test set of manually annotated tweets consisted of 177 negative and 182 positive tweets. In contrast to the training set which was collected based on specific emoticons, the test set was collected by searching Twitter API with specific queries including product names, companies and people. To extend the testing set, we added 641 tweets randomly selected from the original dataset, and annotated manually by 12 users (researchers in our lab), where each tweet was annotated by one user. Our final STS dataset consists of 60K general tweets, with a test set of 1,000 tweets of 527 negatively, and 473 positively annotated ones.

Health Care Reform (HCR)

The Health Care Reform (HCR) dataset was built by crawling tweets containing the hashtag "#hcr" (health care reform) in March 2010 [14]. A subset of this corpus was manually annotated with three polarity labels (*positive, negative, neutral*) and split into training and test sets. In this paper, we focus on identifying positive and negative tweets, and therefore we exclude neutral tweets from this dataset. Identifying neutral tweets is part of our future work plan. The final HCR dataset for training contains 839 tweets, and another 839 tweets were used for testing.

Obama-McCain Debate (OMD)

The Obama-McCain Debate (OMD) dataset was constructed from 3,238 tweets crawled during the first U.S. presidential TV debate in September 2008 [13]. Sentiment ratings of these tweets were acquired using Amazon Mechanical Turk, where each tweet was rated by one or more voter as either *positive, negative, mixed*, or *other*. 'Other" tweets are those that couldn't be rated. We only keep those tweet rated by at least three voters with half of the votes being either *positive* or *negative* to ensure their sentiment polarity. This resulted in a set of 1,081 tweets with 393 positive and 688 negative ones. Due to the relative small size of this dataset, and the lack of a test set, we opted for a 5-fold cross validation approach instead.

4 Semantic Features for Sentiment Analysis

This section describes our semantic features and their incorporation into our sentiment analysis method. As mentioned earlier, the semantic concepts of entities extracted from tweets can be used to measure the overall correlation of a group of entities (e.g. all Apple products) with a given sentiment polarity. Hence adding such features to the analysis could help identifying the sentiment of tweets that contain any of the entities that such concepts represent, even if those entities never appeared in the training set (e.g. a new gadget from Apple).[6]

Semantic features refer to those semantically hidden concepts extracted from tweets [11,12]. An example for using semantic features for sentiment classifier training is shown in Figure 1 where the left box lists entities appeared in the training set together with their occurrence probabilities in positive and negative tweets. For example, the entities "*iPad*", "*iPod*" and "*Mac Book Pro*" appeared more often in tweets of positive polarity and they are all mapped to the semantic concept PRODUCT/APPLE. As a result, the tweet from the test set "*Finally, I got my iPhone. What a product!*" is more likely to have a positive polarity because it contains the entity "*iPhone*" which is also mapped to the concept PRODUCT/APPLE.

4.1 Extracting Semantic Entities and Concepts

There are several open APIs that provide entity extraction services for online textual data. Rizzo and Troncy [10] evaluated the use of five popular entity extraction tools

[6] Assuming of course that the entity extractor successfully identify the new entities as sub-types of concepts already correlated with negative or positive sentiment.

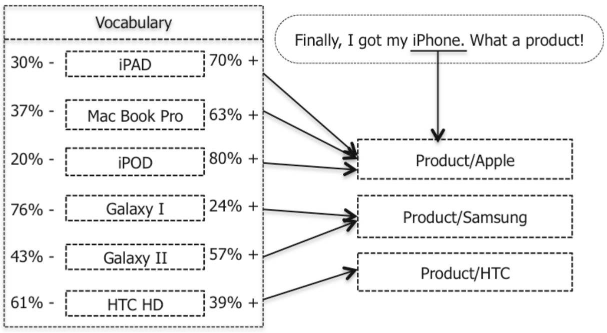


Fig. 1. Measuring correlation of semantic concepts with negative/positive sentiment. These semantic concepts are then incorporated in sentiment classification.

Table 2. Evaluation results of AlchemyAPI, Zemanta and OpenCalais

	No. of Concepts	Entity-Concept Mapping Accuracy (%)		
Extraction Tool	Extracted	Evaluator 1	Evaluator 2	Evaluator 3
AlchemyAPI	108	73.97	73.8	72.8
Zemanta	70	71	71.8	70.4
OpenCalais	65	68	69.1	68.7

on a dataset of news articles, including AlchemyAPI, DBPedia Spotlight,[7] Extractiv,[8] OpenCalais and Zemanta. Their experimental results showed that AlchemyAPI performs best for entity extraction and semantic concept mapping. Our datasets consist of informal tweets, and hence are intrinsically different from those used in [10]. Therefore we conducted our own evaluation, and randomly selected 500 tweets from the STS corpus and asked 3 evaluators to evaluate the semantic concept extraction outputs generated from AlchemyAPI, OpenCalais and Zemanta.

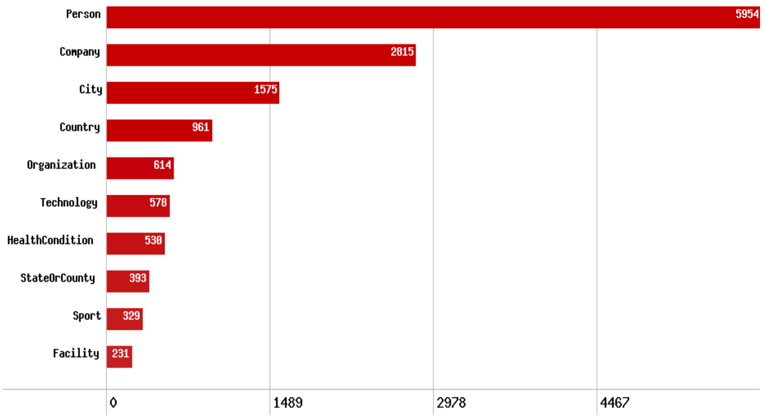
Table 3. Entity/concept extraction statistics of STS, OMD and HCR using AlchemyAPI

	STS	HCR	OMD
No. of Entities	15139	723	1194
No. of Concepts	29	17	14

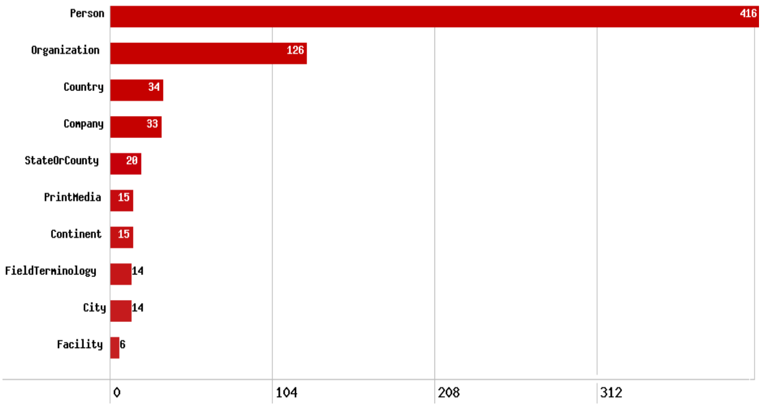
The assessment of the outputs was based on (1) the correctness of the extracted entities; and (2) the correctness of the entity-concept mappings. The evaluation results presented in Table 2 show that AlchemyAPI extracted the most number of concepts and it also has the highest entity-concept mapping accuracy compared to OpenCalais and Zematna. As such, we chose AlchemyAPI to extract the semantic concepts from our three datasets. Table 3 lists the total number of entities extracted and the number of semantic concepts mapped against them for each dataset.

[7] http://dbpedia.org/spotlight/

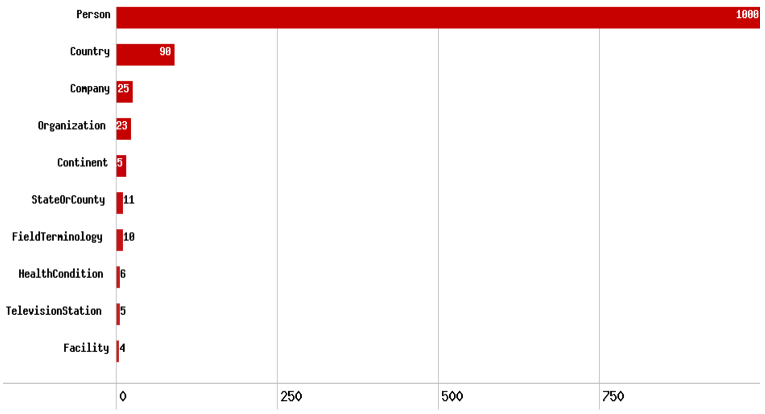
[8] http://wiki.extractiv.com/w/page/29179775/Entity-Extraction



(a) STS.



(b) HCR.



(c) OMD.

Fig. 2. Top 10 frequent concepts extracted with the number of entities associated with them

Figure 2 shows the top ten high-level extracted concepts from the three datasets with the number of entities associated with each of concept. It can be observed that the most frequent semantic concept is PERSON across all the three corpora. The next two most frequent concepts are COMPANY and CITY for STS, ORGANISATION and COUNTRY for HCR, and COUNTRY and COMPANY for OMD. The level of specificity of these concepts is determined by AlchemyAPI.

4.2 Semantic Feature Incorporation

In this section, we propose three different methods to incorporate semantic features into Naive Bayes (NB) classifier training. We start by an overview of the NB followed by our proposed incorporation methods.

NB is a probabilistic classifier, where the assignment of a sentiment class c to a given tweet $\mathbf{w}$ can be computed as:

$$\begin{aligned}\hat{c} &= \arg\max_{c\in\mathcal{C}} P(c|\mathbf{w}) \\ &= \arg\max_{c\in\mathcal{C}} P(c) \prod_{1\leq i\leq N_{\mathbf{w}}} P(w_i|c),\end{aligned} \tag{1}$$

where $N_{\mathbf{w}}$ is the total number of words in tweet $\mathbf{w}$, $P(c)$ is the prior probability of a tweet appearing in class c, $P(w_i|c)$ is the conditional probability of word w_i occurring in a tweet of class c.

In multinomial NB, $P(c)$ can be estimated by $P(c) = N_c/N$ Where N_c is the number of tweets in class c and N is the total number of tweets. $P(w_i|c)$ can be estimated using maximum likelihood with Laplace smoothing:

$$P(w|c) = \frac{N(w,c)+1}{\sum_{w'\in V} N(w'|c) + |V|}, \tag{2}$$

where $N(w,c)$ is the occurrence frequency of word w in all training tweets of class c and $|V|$ is the number of words in the vocabulary.

To incorporate semantic concepts into NB learning, we propose three different methods as described below.

Semantic Replacement: In this method, we replace all entities in tweets with their corresponding semantic concepts. This leads to the reduction of the vocabulary size, where the new size is determined by:

$$|V'| = |V| - |W_{\text{entity}}| + |S|, \tag{3}$$

where $|V'|$ is the new vocabulary size, $|V|$ is the original vocabulary size, $|W_{\text{entity}}|$ is the total number of unique entity words that have been replaced by the semantic concepts, and $|S|$ is the the total number of semantic concepts.

Semantic Augmentation: This method augments the original feature space with the semantic concepts as additional features for the classifier training. The size of the vocabulary in this case is enlarged by the semantic concepts introduced:

$$|V'| = |V| + |S|. \tag{4}$$

Semantic Interpolation: A more principal way to incorporate semantic concepts is through interpolation where we interpolate the unigram language model in NB with the generative model of words given semantic concepts. We propose a general interpolation method below which is able to interpolate arbitrary type of features such as semantic concepts, POS sequences, sentiment-topics etc.

Thus, the new language model with interpolation has the following formula:

$$P_f(W|C) = \alpha P_u(W|C) + \sum_i \beta_i P(W, F_i, C) \tag{5}$$

Where $P_f(W|C)$ is the new language model with interpolation, $P_u(W|C)$ is the original unigram class model and can be calculated using the maximum likelihood estimation, $P(W, F_i, C)$ is the interpolation component, and F_i is a feature vector of type i. The coefficients α and β_i are used to control the influence of the interpolated features in the new language model where:

$$\alpha + \sum_i \beta_i = 1$$

By setting α to 1 the class model becomes a unigram language model without any feature interpolation. On the other hand, setting α to 0 reduces the class model to a feature mapping model. In this work, values of these coefficients have been set by conducting a sensitivity test on the three corpora as will be discuss in Section 6.2.

The interpolation component in the equation 5 can be decomposed as follows:

$$P(W, F_i, C) = \sum_j P(W|f_{ij}) P(f_{ij}|C) \tag{6}$$

Where f_{ij} is the j-th feature of type i, $P(f_{ij}|C)$ is the distribution of features f_{ij} in the training data given the class C and $P(W|f_{ij})$ is the distribution of words in the training data given the feature f_{ij}. Both distributions can be computed via the maximum likelihood estimation.

5 Baselines

We compare the performance of our semantic sentiment analysis approach against the baselines described below.

5.1 Unigrams Features

Word unigrams are the simplest features are being used for sentiment analysis of tweets data. Models trained from word unigrams were shown to outperform random classifiers by a decent margin of 20% [1]. In this work, we use NB classifiers trained from word unigrams as our first baseline model. Table 4 lists, for each dataset, the total number of the extracted unigram features that are used for the classification training.

Table 4. Total number of unigram features extracted from each dataset

Dataset	No. of Unigrams
STS	37054
HCR	2060
OMD	2364

5.2 Part-of-Speech Features

POS features are common features that have been widely used in the literature for the task of Twitter sentiment analysis. In this work, we build various NB classifiers trained using a combination of word unigrams and POS features and use them as baseline models. We extract the POS features using the TweetNLP POS tagger,[9] which is trained specifically from tweets. This differs from the previous work, which relies on POS taggers trained from treebanks in the newswire domain for POS tagging. It was shown that TweetNLP tagger outperforms the Stanford tagger[10] with a relative error reduction of 25% when evaluated on 500 manually annotated tweets [3]. Moreover, the tagger offers additional recognition capabilities for abbreviated phrases, emoticons and interjections (e.g. “lol”, “omg”).

5.3 Sentiment-Topic Features

The sentiment-topic features are extracted from tweets using the weakly-supervised joint sentiment-topic (JST) mode that we developed earlier [7]. We trained this model on the training set with tweet sentiment labels discarded. The resulting model assigns each word in tweets with a sentiment label and a topic label. Hence JST essentially groups different words that share similar sentiment and topic.

We list some of the topic words extracted by this model from the STS and OMD corpora in Table 5. Words in each cell are grouped under one topic and the upper half of the table shows topic words bearing positive sentiment while the lower half shows topic words bearing negative sentiment. For example, Topic 2 under positive sentiment is about the movie “Twilight”, while Topic 5 under negative sentiment is about a complaint of feeling sick possibly due to cold and headache. The rational behind this model is that grouping words under the same topic and bearing similar sentiment could reduce data sparseness in Twitter sentiment classification and improves accuracy.

6 Evaluation Results

In this section, we evaluate the use of the sentiment features discussed in 4 and present the sentiment identification results on the STS, HCR and OMD datasets. We then compare these results with those obtained from using the baseline features described in Section 5.

[9] http://www.ark.cs.cmu.edu/TweetNLP/

[10] http://nlp.stanford.edu/software/tagger.shtml

Table 5. Extracted sentiment-topic words by the sentiment-topic model

	Topic 1	Topic 2	Topic 3	Topic 4	Topic 5
Positive	win	twilight	make	tomorrow	today
	final	movie	mccain	weekend	nice
	watch	award	debate	school	Sunday
	game	moon	good	start	enjoy
	luck	tonight	point	plan	weather
	today	watch	interest	fun	love
	week	mtv	right	yai	walk
	hope	excited	answer	wait	sunny
Negative	iphone	dog	obama	miss	feel
	internet	sad	question	far	sick
	download	death	understand	travel	bad
	apple	accident	doesn't	mum	hurt
	store	today	answer	away	pain
	slow	car	comment	dad	flu
	issue	awful	back	love	sore
	crash	cry	debate	country	horrible

We use NB trained from word unigrams as the starting-point baseline model. The features are incorporated into NB by either the interpolation approach described in Section 4.2 or by simply augmenting into the original bag-of-words feature space. For evaluation on STS and HCR, we use the training and testing sets shown in Table 1. For OMD, we perform 5-fold cross validation and report the results averaged over 10 such runs.

The raw tweets data can be very noisy, and hence some pre-processing was necessary, such as replacing all hyperlinks with "*URL*", converting some words with apostrophe, such as "*hate'n*", to their complete form "*hating*", removing repeated letters (e.g. "*loovee*" becomes "*love*"), etc.

6.1 Stopwords

It is a common practice to perform stopwords removal as a standard pre-processing step by removing those common words which tend to have little meaning. Nevertheless, Bei [16] argued that stopwords can be used as discriminative features for specific classification tasks. We have conducted a set of experiments to evaluate the impact of stopwords removal on sentiment classification on tweets. We compare the performance of a NB classifier trained from word unigrams before and after removing the stopwords. It can be observed from Table 6 that the classifiers learned with stopwords outperform those learned with stopwords removed. Similar outcome was observed when using all out sentiment analysis features. Hence, we chose to keep the stopwords in our subsequent experiments.

6.2 Incorporating Semantic Features

Semantic features can be incorporated into NB training in three different ways, *replacement*, *augmentation*, and *interpolation* (Section 4.2). Table 7 shows the F measures

Table 6. Sentiment classification accuracy ((True Positives + True Negatives) / Total) with and without stopwords using unigram features

Dataset	With Stopwords	Without Stopwords
Stanford Twitter Sentiment (STS)	**80.7%**	77.5%
Health Care Reform (HCR)	**71.1%**	68.5%
Obama-McCain Debate (OMD)	**75.4%**	73.7%

produced when using each of these feature incorporation methods. With *semantic replacement*, where all entities in tweets are *replaced* with their corresponding semantic concepts, the feature space shrunk substantially by nearly 15-20%, and produced an average F measure of 68.9%. However, this accuracy is 3.5% and 10.2% less than when using semantic augmentation and interpolation respectively. The performance degradation is due to the information loss caused by this term replacement which subsequently hurts NB performance.

Augmenting the original feature space with semantic concepts (*semantic augmentation*) performs slightly better than *sentiment replacement*, though it still performs 6.5% worse than interpolation. With *Semantic interpolation*, semantic concepts are incorporated into NB training taking into account the generative probability of words given concepts. This method produces the highest accuracy amongst all three incorporation methods, with an average F of 75.95%.

Table 7. Average sentiment classification accuracy (%) using different methods for incorporating the semantic features. Accuracy here is the average harmonic mean (F measure) obtained from identifying positive and negative sentiment.

Method	STS	HCR	OMD	Average
Semantic replacement	74.10	61.35	71.25	68.90
Semantic augmentation	77.65	63.65	72.70	71.33
Semantic interpolation	**83.90**	**66.10**	**77.85**	**75.95**

The contribution of semantic features in the interpolation model is controlled by the interpolation coefficients in Equation 5. We conducted a sensitivity test to evaluate the impact of the interpolation coefficients on sentiment classification accuracy by varying β between 0 and 1. Figure 3 shows that accuracy reaches its peak with β set between 0.3 and 0.5. In our evaluation, we used 0.4 for STS dataset, and 0.3 for the other two.

6.3 Comparison of Results

In this section we will compare the Precision, Recall, and F measure of our semantic sentiment analysis against the baselines described in Section 5. We report the semantic classification results for identifying positive and negative sentiment separately to allow for deeper analysis of results. This is especially important given how some analysis methods perform better in one sentiment polarity than in the other.

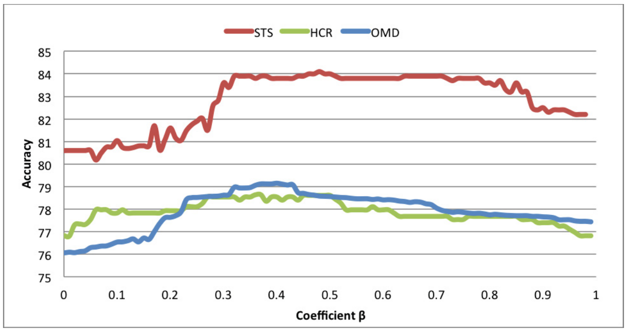


Fig. 3. Sensitivity test of the interpolation coefficient for semantic interpolation

Table 8 shows the results of our sentiment classification using *Unigrams*, *POS*, *Sentiment-Topic*, and *Semantic* features, applied over the STS, HCR, and OMD datasets which are detailed in Section 3. The table reports three sets of P, R, and F1, one for positive sentiment identification, one for negative sentiment identification, and the third shows the averages of the two.

Table 8. Cross comparison results of all the four features

Dataset	**Feature**	**Positive Sentiment**			**Negative Sentiment**			**Average**		
		P	R	F1	P	R	F1	P	R	F1
STS	Unigrams	82.20	75.20	78.50	79.30	85.30	82.20	80.75	80.25	80.35
	POS	83.70	75.00	79.10	79.50	86.90	83.00	81.60	80.95	81.05
	Sentiment-Topic	80.70	82.20	81.40	83.70	82.30	83.00	82.20	82.25	82.20
	Semantics	85.80	79.40	82.50	82.70	88.20	85.30	**84.25**	**83.80**	**83.90**
HCR	Unigrams	39.00	36.10	37.50	81.00	82.80	81.90	60.00	59.45	59.70
	POS	56.20	22.00	31.70	80.00	94.70	86.70	68.10	58.35	59.20
	Sentiment-Topic	53.80	47.20	50.30	84.50	87.60	86.00	**69.15**	**67.40**	**68.15**
	Semantics	53.60	40.40	46.10	83.10	89.30	86.10	68.35	64.85	66.10
OMD	Unigrams	64.20	70.90	67.10	83.30	78.60	80.80	73.75	74.75	73.95
	POS	69.50	68.30	68.70	83.10	83.90	83.40	76.30	76.10	76.05
	Sentiment-Topic	68.20	75.60	71.70	87.10	82.40	84.70	77.65	**79.00**	**78.20**
	Semantics	75.00	66.60	70.30	82.90	88.10	85.40	**78.95**	77.35	77.85

According to these results in Table 8, the Semantic approach outperforms the Unigrams and POS baselines in all categories and for all three datasets. However, for the HCR and OMD datasets, the sentiment-topic analysis approach seem to outperform the semantic approach by a small margin. For example, the semantic approach produced higher P, R, and F1 for the STS dataset, with F1 4.4% higher than Unigrams, 3.5% higher than POS, and 2.1% higher than the sentiment-topic features. In HCR, F1 from the semantic features were 8.9% and 11.7% higher than Unigrams and POS, but 3% lower than F1 from sentiment-topic features. For OMD, semantic features also outperformed the Unigrams and POS baselines, with 5.2% and 2.4% higher F1 respectively.

However, in the OMD dataset, F1 from semantic features was 0.4% lower than from the topic model, although Precision was actually higher by 1.7%.

As detailed in Section 3 and Table 1, the STS dataset consists of a large collection of general tweets with no particular topic focus. Unlike STS, the other two datasets are much smaller in size and their tweets discuss very specific topics; the US Health Care Reform bill in the HCR dataset, and the Obama McCain debate in the OMD dataset. Using semantic features seem to perform best in the large and general dataset, whereas the sentiment-topic features seem to take the lead in small, topic-focused datasets. The reason is likely to be that classifying with sentiment-topic features group words into a number of topics. In our experiments, we found that for the STS dataset, increasing the number of topics leads to the increase of classification accuracy with the peak value of 82.2% reached at topic number 50. Further increasing topic numbers degrades the classifier performance. However, for HCR and OMD, the best accuracy was obtained with only one topic (68.15% for HCR and 78.20% for OMD). The classification performance drops significantly by any further increment. This can be explained by the nature of these three datasets. HCR was collected using the hashtag "#hcr" (health care reform) while OMD consists of tweets about the Obama-McCain debate. Hence these two datasets are topic-specific. On the contrary, STS was collected using more general queries and thus it contains a potentially large number of topics.

Hence the benefits of using the sentiment-topic features seem to be reduced in comparison to semantic features when the training set is of general content as in the STS tweets dataset.

The average results across all three datasets are shown in Table 9. Here we can see that semantic features do better than sentiment-topic features and the other baselines when identifying *negative sentiment*. However, sentiment-topic features seem to perform better for *positive sentiment*. For positive sentiment, using the semantic approach produces Precision that is better than Unigrams, POS, and sentiment-topic by 15.6%, 2.4%, and 5.8% respectively. However, the Recall produced by the semantic approach when identifying positive sentiment is 2.3% and 12.8% higher than in Unigrams and POS, but 9% lower than Recall from the sentiment-topic approach. Overall, F for positive sentiment from semantic features is 2.2% lower than when using sentiment-topic features. It is worth emphasising that the average Precision from identifying both positive and negative sentiment is the highest at 77.18% when using semantic features. When analysing large amounts of continuously flowing data as with social media resources, Precision could well be regarded as much more important than Recall.

Table 9. Averages of Precision, Recall, and F measures across all three datasets

Features	**Positive Sentiment**			**Negative Sentiment**			**Average**		
	P	R	F1	P	R	F1	P	R	F1
Unigrams	61.80	60.73	61.03	81.20	82.23	81.63	71.50	71.48	71.33
POS	69.80	55.10	59.83	80.87	88.50	84.37	75.53	72.23	72.48
Sentiment-Topic	67.57	**68.33**	**67.80**	**85.10**	84.10	84.57	77.02	**76.73**	**76.75**
Semantics	**71.47**	62.13	66.30	82.90	**88.53**	**85.60**	**77.18**	75.33	75.95

7 Discussion and Future Work

In this paper we demonstrated the value of using semantic features for the classification of positive and negative sentiment in Tweets. We tested several off-the-shelf semantic entity extractors and decided on using AlchemyAPI due to its better performance in terms of coverage and accuracy. One thing that impacts our results is the abstraction level of the concepts retrieved from the entity extractor. In many cases, these concepts were too abstract (e.g. Person) which were equally used for mentions of ordinary people, as well as for famous musicians or politicians. For the tweet "i wish i could go to france and meet president Obama haha", AlchemyAPI provided the concept *Person* to represent "president Obama", whereas Zemanta identified him with the concept */government/politician* which is more specific. In future work we plan to devise an approach to increase the specificity of such concepts, perhaps with the aid of DBpedia or using multiple entity extractors and comparing the specificity level of their proposed concepts.

In our evaluation of AlchemyAPI, Zemanta, and OpenCalais, we observed that some of them perform better than others for specific type of entities. For example, Zemanta produced more accurate and specific concepts to describe entities related to music tracks and bands. It might be possible to implement a more selective approach, where certain semantic extractors and concept identifiers are used, or trusted more, for certain type of entities.

When using semantic features, all identified concepts in a tweet are added to the analysis. However, it might be the case that semantic features improve sentiment analysis accuracy for some type of concepts (e.g. cities, music) but reduce accuracy in some other concept types (e.g. people, companies). We will investigate the impact of each group of concepts on our analysis accuracy, to determine their individual contribution and impact on our sentiment analysis. We can also assign weights to each concept type to represent its correlation with positive or negative sentiment.

We experimented with multiple datasets of varying sizes and topical-focus. Our results showed that the accuracy of classifying with some feature selections can be sensitive to the size of the datasets and their topical-focus. For example, our evaluation showed that the semantic approach excels when the dataset is large and of diverse topic coverage. In future work we will apply these approaches on larger datasets to examine the consistency of their performance patterns. Furthermore, we also intend to explore various feature selection strategies to improve the sentiment classification performance.

Our sentiment analysis focused on positive and negative tweets. Neutral sentiment tend to be much harder to identify as it requires the determination of the context of the tweet message. For example, some words of a tweet may have both subjective and objective senses. Handling such tweets will therefore require the introduction of another classifier, to identify subjective/objective tweets.

8 Conclusions

We proposed the use of semantic features in Twitter sentiment classification and explored three different approaches for incorporating them into the analysis; with replacement, augmentation, and interpolation. We found that best results are achieved when interpolating the generative model of words given semantic concepts into the unigram language model of the NB classifier. We conducted extensive experiments on three Twitter datasets and compared the semantic features with the the Unigrams and POS sequence features as well as with the sentiment-topic features. Our results show that the semantic feature model outperforms the Unigram and POS baseline for identifying both negative and positive sentiment. We demonstrated that adding semantic features produces higher Recall and F1 score, but lower Precision, than sentiment-topic features when classifying negative sentiment. We also showed that using semantic features outperforms the sentiment-topic features for positive sentiment classification in terms of Precision, but not in terms of Recall and F1. One average, the semantic features appeared to be the most precise amongst the four other feature selections we experimented with.

Our results indicates that the semantic approach is more appropriate when the datasets being analysed are large and cover a wide range of topics, whereas the sentiment-topic approach was most suitable for relatively small datasets with specific topical foci.

We believe that our findings demonstrated the high potential of the novel approach of interpolating semantic features into the sentiment classifier. In our current implementation, we rely on Alchemy API which is only able to produce rather coarse semantic concept mappings. However, our results indicate that further gains could be achieved when entities are mapped into a more fine-grained semantic concept space.

Acknowledgment. The work of the authors was supported by the EU-FP7 project ROBUST (grant no. 257859).

References

1. Agarwal, A., Xie, B., Vovsha, I., Rambow, O., Passonneau, R.: Sentiment analysis of twitter data. In: Proc. ACL 2011 Workshop on Languages in Social Media, pp. 30–38 (2011)
2. Barbosa, L., Feng, J.: Robust sentiment detection on twitter from biased and noisy data. In: Proceedings of COLING, pp. 36–44 (2010)
3. Gimpel, K., Schneider, N., O'Connor, B., Das, D., Mills, D., Eisenstein, J., Heilman, M., Yogatama, D., Flanigan, J., Smith, N.: Part-of-speech tagging for twitter: Annotation, features, and experiments. Tech. rep., DTIC Document (2010)
4. Go, A., Bhayani, R., Huang, L.: Twitter sentiment classification using distant supervision. CS224N Project Report, Stanford (2009)
5. Guerra, P., Veloso, A., Meira Jr., W., Almeida, V.: From bias to opinion: A transfer-learning approach to real-time sentiment analysis. In: Proceedings of the 17th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD (2011)
6. Kouloumpis, E., Wilson, T., Moore, J.: Twitter sentiment analysis: The good the bad and the omg! In: Proceedings of the ICWSM (2011)
7. Lin, C., He, Y.: Joint sentiment/topic model for sentiment analysis. In: Proceeding of the 18th ACM Conference on Information and Knowledge Management, pp. 375–384. ACM (2009)
8. Moschitti, A.: Efficient Convolution Kernels for Dependency and Constituent Syntactic Trees. In: Fürnkranz, J., Scheffer, T., Spiliopoulou, M. (eds.) ECML 2006. LNCS (LNAI), vol. 4212, pp. 318–329. Springer, Heidelberg (2006)
9. Pak, A., Paroubek, P.: Twitter as a corpus for sentiment analysis and opinion mining. In: Proceedings of LREC 2010 (2010)

10. Rizzo, G., Troncy, R.: Nerd: Evaluating named entity recognition tools in the web of data. In: Workshop on Web Scale Knowledge Extraction (WEKEX 2011), vol. 21 (2011)
11. Saif, H., He, Y., Alani, H.: Semantic Smoothing for Twitter Sentiment Analysis. In: Proceeding of the 10th International Semantic Web Conference, ISWC (2011)
12. Saif, H., He, Y., Alani, H.: Alleviating Data Sparsity for Twitter Sentiment Analysis. In: Proceedings, 2nd Workshop on Making Sense of Microposts (#MSM 2012): Big Things Come in Small Packages: in Conjunction with WWW (2012)
13. Shamma, D., Kennedy, L., Churchill, E.: Tweet the debates: understanding community annotation of uncollected sources. In: Proceedings of the First SIGMM Workshop on Social Media, pp. 3–10. ACM (2009)
14. Speriosu, M., Sudan, N., Upadhyay, S., Baldridge, J.: Twitter polarity classification with label propagation over lexical links and the follower graph. In: Proceedings of the EMNLP First Workshop on Unsupervised Learning in NLP, pp. 53–63 (2011)
15. Tan, C., Lee, L., Tang, J., Jiang, L., Zhou, M., Li, P.: User-level sentiment analysis incorporating social networks. In: Proceedings of the 17th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD (2011)
16. Yu, B.: An evaluation of text classification methods for literary study. Literary and Linguistic Computing 23(3), 327–343 (2008)

CrowdMap: Crowdsourcing Ontology Alignment with Microtasks

Cristina Sarasua[1], Elena Simperl[1], and Natalya F. Noy[2]

[1] Institute AIFB. Karlsruhe Institute of Technology
csarasuagar@gmail.com, elena.simperl@kit.edu
[2] Stanford University
noy@stanford.edu

Abstract. The last decade of research in ontology alignment has brought a variety of computational techniques to discover correspondences between ontologies. While the accuracy of automatic approaches has continuously improved, human contributions remain a key ingredient of the process: this input serves as a valuable source of domain knowledge that is used to train the algorithms and to validate and augment automatically computed alignments. In this paper, we introduce CrowdMap, a model to acquire such human contributions via microtask crowdsourcing. For a given pair of ontologies, CrowdMap translates the alignment problem into microtasks that address individual alignment questions, publishes the microtasks on an online labor market, and evaluates the quality of the results obtained from the crowd. We evaluated the current implementation of CrowdMap in a series of experiments using ontologies and reference alignments from the Ontology Alignment Evaluation Initiative and the crowdsourcing platform CrowdFlower. The experiments clearly demonstrated that the overall approach is feasible, and can improve the accuracy of existing ontology alignment solutions in a fast, scalable, and cost-effective manner.

1 Introduction

The last decade of research on ontology alignment has brought a wide variety of automatic methods and techniques to discover correspondences between ontologies. Researchers have studied extensively the strengths and weaknesses of existing solutions, as well as their natural limitations and principled combinations, not least through community projects such as the Ontology Alignment Evaluation Initiative (OAEI).[1] Partly as a result of these efforts the performance of the underlying algorithms has continuously improved. However, most researchers believe that human assistance is nevertheless required, even if it is just for the validation of automatically computed mappings. In this paper, we introduce CrowdMap an approach to integrate human and computational intelligence in ontology alignment tasks via microtask crowdsourcing.

The term "microtask crowdsourcing" refers to a problem-solving model in which a problem is outsourced to a distributed group of people by splitting the problem space into smaller sub-problems, or tasks, that multiple workers address independently in

[1] http://oaei.ontologymatching.org/

return for a (financial) reward. Probably the most popular online instantiation of this model is Amazon's Mechanical Turk (MTurk) platform (https://www.mturk.com/) which offers a virtual labor marketplace for microtasks as well as basic services for task design and publication, work assignment, and payment. Typical problems that are amenable to microtask crowdsourcing are those problems that we can easily distribute into a (high) number of simple tasks, which workers can complete in parallel, in a relatively short period of time (in the range of seconds to minutes), and without specific skills or expertise. Examples of such problems include finding a specific piece of information on the Web, labeling or classifying content, and ranking a list of objects. Recently, researchers have demonstrated the effectiveness of microtask crowdsourcing for far more complex problems by using sophisticated workflow management techniques on top of the basic services of existing platforms, and optimizing quality assurance and work assignment [1,2,3]. As a result, microtask crowdsourcing has been successfully applied to a broad range of diverse problems: completing surveys, translating text from one language to another, creating comprehensive product descriptions, matching pictures of people, summarizing text [4] and many others.

Ontology alignment is a good fit for microtask crowdsourcing for several reasons. First, verifying whether or not a mapping is a correct one is naturally a microtask, and workers do not need much context to figure out the right answer. Second, we can easily decompose the overall problem of verification of a set of candidate mappings into atomic tasks corresponding to the individual mappings. These tasks are largely independent of one another. Third, while ontologies can be quite large (with tens of thousands of classes), their scale is often considerably smaller than the scale of the data itself. Thus, crowdsourcing becomes a tractable way to verify all candidate alignments between two ontologies. Finally, ontology alignment is still one of those problems that we cannot automate completely, and having a human in the loop might increase the quality of the results of machine-driven approaches.

There are two different ends of the spectrum in which we envision applying crowdsourcing to ontology alignment. On the one hand, we can generate all possible pairs of alignments between two ontologies, and ask the crowd to evaluate each of the candidates. However, this option will clearly not scale well, as we will be asking the users to inspect an extremely large number of pairs—equivalent to the cartesian product of the size of the two ontologies—and we know that the number of valid correspondences are usually at most comparable to the number of terms in the smaller of the two ontologies. On the other hand, we can start by running an automatic algorithm that generates potential alignments, and subsequently have the crowd assess the results. This second option will likely be much more scalable in terms of the number of tasks and answers needed from the crowd (and thus the duration and cost of the alignment exercise). While this scenario is likely to lead to improvements in the precision of the original algorithm, with this approach we will be able to have similar effects also on the recall if we present the crowd with the very low confidence mappings.

CROWDMAP is a new model for ontology alignment which uses microtask crowdsourcing to improve the accuracy of existing automatic solutions. In evaluating this approach, we explore the following research questions:

R1 Is ontology alignment amenable to microtask crowdsourcing?
R2 How does such a human-driven approach compare with automatic (or semi-automatic) methods and techniques, and can it improve their results?
R3 What types of alignment problems can workers feasibly solve? What correspondences between elements of different ontologies (e.g., similar, more general, more specific) can be reliably identified via crowdsourcing?

We introduce CROWDMAP and its implementation using CrowdFlower (`http://crowdflower.com/`) a crowdsourcing platform which acts as an intermediary to a number of online labor marketplaces, including MTurk. For a given pair of ontologies, CROWDMAP translates the alignment problem into microtasks that address individual alignment questions, publishes the microtasks on an online labor market, and evaluates the quality of the results obtained from the crowd. We tested the current implementation in multiple settings in order to determine how we can optimize the quality of the crowdsourced results through specific task-design and work-assignment features. For this purpose we ran a series of different experiments: an exhaustive alignment between two (smaller) ontologies; a broader set of ontologies assessing the outcomes produced by a simulated automatic algorithm; and validating the mappings computed by one of the algorithms that participated in Ontology Alignment Evaluation Initiative. The experiments provided evidence that the overall idea to apply microtask crowdsourcing to ontology alignment is not only feasible, but can also significantly improve the precision of existing ontology alignment solutions in a fast, scalable, and cost-effective manner. The findings of the experiments allowed us to define a number of best practices for designing purposeful ontology alignment projects, in which human and computational intelligence are smoothly interwoven and yield better results in terms of costs and quality compared to state-of-the-art automatic or semi-automatic approaches.

2 Related Work

While the ontology alignment community acknowledges the importance of human contributions, the question of how to optimally collect and harvest these contributions leaves room for further research [5]. Falconer and colleagues described the results of an observational study of the problems users experience when aligning ontologies [6]. They emphasized the difficulties experienced by laymen in understanding and following the individual steps of an alignment algorithm. In our work, we provide further evidence for the extent to which contributions from non-technical users can provide valuable input in the alignment process, and investigate alternative means to describe and document alignment tasks in order to make them accessible to laymen.

Another approach employs Web 2.0 technologies and principles to engage a community of practice in defining alignments, thus increasing the acceptance of the results, and reducing or distributing the associated labor costs [7,8,9,10]. An early proposal on collaborative ontology alignment by Zhdanova and Shvaiko [10] developed a community-driven service that allowed users to share alignments in a publicly available repository. BioPortal [11] offers a comprehensive solution in the biomedical domain. It enables users to create alignments between individual elements of an ontology [9]. However, in these approaches, the solicitation for the mappings is "passive": the users must come to the site, find the terms of interest, and create the mappings. There is no expected reward, other than community recognition. By contrast, our CrowdMap model is essentially "mapping for hire" where we do not expect users to have a specific interest in the task that they perform other than the monetary reward that they get. Our experience shows that there is no comparison in the quantity of the work that can be obtained via volunteering and microtask crowdsourcing: putting aside the different knowledge domains that the two approaches address, we were able to get orders of magnitude more alignments in a day in the experiments with the current CrowdMap implementation than BioPortal received in a year. In this paper, we evaluate the quality of these mappings to determine how useful the microtask-based alternative is beyond the actual number of mappings generated.

McCann and colleagues studied motivators and incentives in ontology alignment [7]. They investigated a combination of volunteer and paid user involvement to validate automatically generated alignments formulated as natural-language questions. While this proposal shares many commonalities with CrowdMap, the evaluation of their solution is based on a much more constrained experiment that did not rely on a real-world labor marketplace and associated work force.

Games with a purpose, which capitalize on entertainment, intellectual challenge, competition, and reputation, offer another mechanism to engage with a broad user base. In the field of semantic technologies, the OntoGame series proposes several games that deal with the task of data interlinking, be that in its ontology alignment instance (SpotTheLink [12]) or multimedia interlinking (SeaFish [13]). Similar ideas are implemented in GuessWhat?!, a selection-agreement game which uses URIs from DBpedia, Freebase and OpenCyc as input to the interlinking process [14]. While OntoGame looks into game mechanics and game narratives and their applicability to finding similar entities and other types of correspondences, our research studies an alternative crowdsourcing strategy that is based on financial rewards in a microtask platform.

More recently, researchers in the Semantic Web community have begun to explore the feasibility of crowdsourcing for assigning URIs to entities that are discovered in textual Web pages. ZenCrowd, for example, combines the results of automatically and human-generated answers to link entities recognized in a text with entities in the Linked Open Data cloud [15]. ZenCrowd developers proposed a variety of techniques to reduce the scope of the crowdsourcing task, such as excluding candidates for which an algorithm already has a high confidence score from the set to be validated. Our approaches are similar in spirit (using the crowd to improve the performance of automatic algorithm in alignment). However, ontology alignment (rather than data alignment) has a more tractable scope. The motivation of our work is also different: our goal is not to identify which of the two approaches (machine vs human-driven) are likely to be more reliable, but to enhance the results produced by an automatic algorithm.

3 The CrowdMap Definition and Implementation

CrowdMap takes as input a set of *candidate mappings* between two ontologies and uses a *microtask platform* to improve their accuracy. The model is not bound to a specific instantiation of microtask platform. It can be applied to any virtual labor marketplace that enables requesters to post a problem as a set of independent *microtasks*, which are performed in parallel by *workers* in return for a (usually monetary) reward. In fact, we can apply the same model to other approaches to human computation, such as games with a purpose, which, though operating on different motivational factors, address similar types of problems: decomposable, verifiable, and not requiring domain-specific knowledge or skills.

3.1 Fundamentals of Microtask Crowdsourcing

In order to use a microtask platform, a requester packages the work into microtasks and publishes them in batches or groups. Amazon Mechanical Turk (MTurk), one of the most popular crowdsourcing platforms, refers to microtasks as *Human Intelligence Tasks (HITs)*, a term that we will use interchangeably with microtask.

A requester specifies a number of configuration parameters such as the number of answers that she needs for each HIT, the time to complete a HIT, and restrictions on the profile of the workers (e.g., geographical location, knowledge of a specific natural language). As most HITs can be solved quickly (within seconds or minutes at most), similar HITs are typically organized into groups or batches which share the same configuration parameters; workers prefer to be assigned to such larger chunks of work instead of dealing with atomic questions in separate processes. Upon completion of the tasks by workers, the requester collects and assesses the responses and rewards the accepted ones according to the pre-defined remuneration scheme. For most platforms, the requester can automate the interaction with the system via an API, while the workers undertake their tasks using a Web-based interface generated by the requester. The overall effectiveness of crowdsourcing can be influenced dramatically by the way that the requester packages a given problem as a series of microtasks [16,17]. This packaging includes, in particular, the design of the interface (including clear instructions for the completion of the task, minimal quality criteria for the work to be accepted, and purposeful layout), and the procedures that the requester uses in order to evaluate the results and to measure the performance of workers. Because multiple workers can perform the same microtask, the requester can implement different types of quality assurance [1]. For example, one can use majority voting (take the solution on which the majority of workers agree), or more sophisticated techniques that take into account, for instance, the (estimated) expertise of specific workers, or the probabilistic distribution of accuracy of the answers of a given worker. In addition, the requester needs to implement mechanisms to avoid and detect spam in order to reduce the overhead associated with the evaluation of the crowd-produced results. Other factors that are proven to influence the success of crowdsourcing (in particular in terms of the duration of the execution of the tasks, and the ability to find appropriate work resources in due time) are the number of HITs per batch, and the frequency of publication of similar HITs groups, and the novelty of the tasks. Studies showed that whereas grouping HITs into batches leads to economies of scale, batches of several hundreds of HITs are more difficult to assign than the ones with a size up to 100 questions [17]. An analogously motivated behavior of workers tending to focus their resources on similarly scoped tasks makes finding assignments for larger problems divided into several batches and HITs more challenging, as finding different eligible workers in due time to address the entire body of work becomes more difficult. Researchers have studied ways to expand the original application scope of MTurk and alike to more complex workflows [3], problems with an open, unknown set of solutions [4], or those characterized by tight time-to-completion constraints [18].

CrowdMap uses CrowdFlower, one of the leading crowdsourcing platforms as a basis for its implementation. CrowdFlower is an intermediary: it is not itself an online labor market, but it publishes microtasks to different crowds simultaneously (including MTurk, Crowd Guru, getpaid, Snapvertise, and others). It implements advanced quality assurance methods based on golden standards in addition to the basic functionality of the crowdsourcing platforms that it accesses. Specifically, CrowdFlower uses "golden units" to denote those types of alignment questions, for which the answer is trivial or known in advance. CrowdMap evaluates whether or not a worker can be trusted by extrapolating from the accuracy of the answers she gave to these particular questions. These methods help determine the reliability and performance of workers, and to filter spammers at run time [19]. The terminology used by CrowdFlower to denominate the core concepts of microtask crowdsourcing is slightly different than the one adopted by MTurk. HITs or microtasks are termed "jobs", and answers (or "assignments" in MTurk) to these questions are "judgements". HITs become "job assignments" in CrowdFlower when they are built using job templates and particular data "units". MTurk organizes CrowdFlower tasks in "batches", when they share the title. In the remainder of the paper we will use these terms interchangeably.

3.2 The CrowdMap Workflow

The CrowdMap task is to find a set of mappings between two ontologies, O_1 and O_2. First, an automatic mapping algorithm A produces a set of candidate mappings between O_1 and O_2. Each candidate mapping m represents a potential correspondence between a concept in O_1 and a concept in O_2. The concepts can be classes, properties, or axioms in the ontologies. Correspondences are typically an equivalence or a similarity relation ($=$), but can be a subsumption relation ($<=$, $>=$), or any other (domain-specific) relation. In the current implementation of CrowdMap, we consider only $=$, $<=$, and $>=$. The algorithm A may also produce a confidence measure $conf$. If A does not produce confidence measures, then we assume that $conf = 1$ for all mappings returned by A.

We generate microtasks as follows.

– There is a microtask to verify each candidate mapping m. Tasks can either ask workers either to validate a given mapping relationship between the source and target (such as similarity), or to choose between different types of relationships between the source and the target (such as subsumption, similarity, or meronymy).
– If the algorithm A produces only equivalence (similarity) mappings, then CrowdMap requests 3 workers to verify the same mapping.
– If the algorithm A produces equivalence and subsumption mappings, then CrowdMap asks for up to 7 workers to complete the task of selecting a relationship between the source and target, until at least two of them agree on a choice of relationship between the two terms.

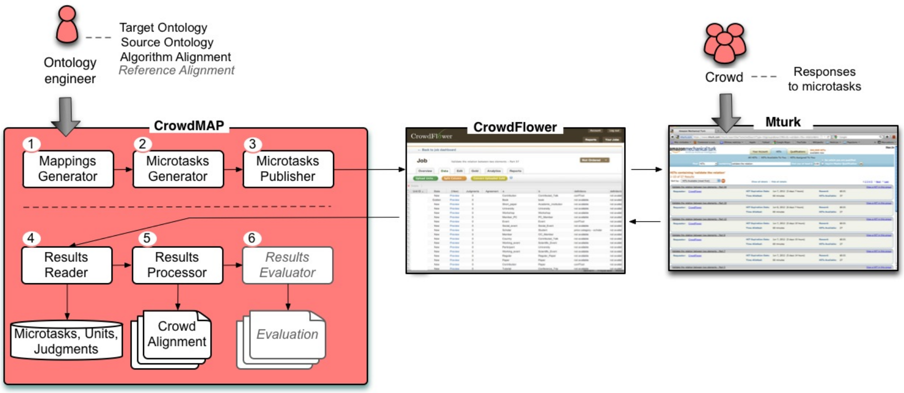


Fig. 1. CROWDMAP architecture. CROWDMAP generates microtasks using a set of pairs of ontological elements and the relationships between them, publishes the microtasks to CrowdFlower, retrieves the answers of the crowd, and compiles the final alignment results by deciding which of these answers are valid.

– The final set of mappings is the set of mappings M_c where at least 2 workers agreed on the type of the mapping.

The number of workers that we assign for each microtask is a configuration parameter. The values that we used in the current version of CROWDMAP follow common practice in using microtask platforms for similar types of tasks. We assume that a higher number of answers are required to validate the second type of task (asking for equivalence and subsumption), which is significantly more complex from an alignment point of view and has more options for workers to choose from. Our pilot studies helped us determine others, such as the choice of words and methods to avoid spam (Section 5).

3.3 The CROWDMAP Architecture

Figure 1 shows the CROWDMAP architecture. The dashed line separates the modules that prepare and publish microtasks from the modules that process the responses of the crowd. CROWDMAP executes the former set of modules first (see the specific order in the numbers). Once CROWDMAP creates the microtasks in CrowdFlower and they are published to the actual labor platforms such as MTurk, the crowd interacts with the MTurk interface and provides responses to the microtasks. When CrowdFlower receives the full set of answers for these microtasks, CROWDMAP executes the second set of modules and calculate the resulting alignment.

Mappings Generator. The current CROWDMAP prototype focuses of pairs of classes as elements to be compared through crowdsourced alignment. We do not yet support mappings between properties, but many of the main findings of our experiments are likely to apply to these types of ontological primitives as well. The Mappings Generator processes the alignment from an automatic tool or uses one of its benchmark-generation mechanisms to generate a set of mappings to test. Section 4 discusses the different sets of candidate mappings that we generated for the experiments.

Microtasks Generator. This module generates the microtasks associated with the pairs of classes computed by the Mappings Generator. We can further parameterize the process by configuring such aspects as interface and layout, number of answers for each alignment question, number of questions within one microtask, and restrictions on the workforce (e.g., a certain level of performance achieved so far, geo-location, language skills). The result is the actual interface that the workers will use in order to submit their answers.

Microtasks Publisher. The publisher module posts the microtasks to the crowdsourcing platform. In the current implementation we support the publication to CrowdFlower using the API that it provides. The publisher module creates the corresponding microtasks on CrowdFlower, uploads the data about the normal and the golden units, and publishes the microtasks on MTurk.

Results Reader. Once the microtasks are completed, CrowdFlower calculates an aggregated response for each pair of terms to align, as well as the confidence of such aggregated responses. The confidence combines the accuracy that workers obtained in the microtask with the agreement of the responses for the alignment question at hand. Access to this information is provided through the CrowdFlower API.

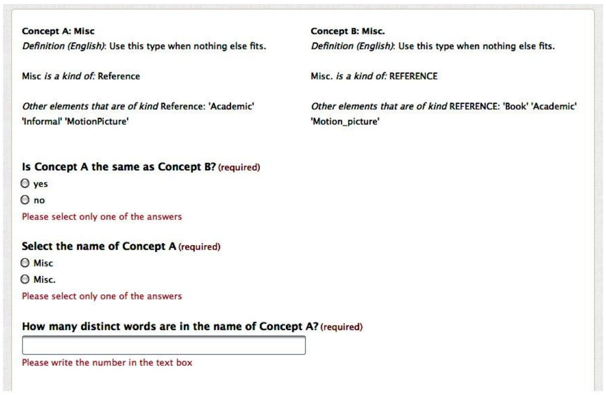
Results Processor. This module generates a file with the crowd alignment, serialized in the Alignment API format [20]. The usage of this standard format facilitates the comparison between different approaches (crowdsourced vs. automatic, reference data vs. manually or automatically generated), as well as the reuse of the results in new scenarios involving both human-oriented and algorithmic processing.

Results Evaluator. The evaluator module relies on the Alignment API to assess the crowd alignment. Via the API we access information about specific alignments (the ones computed by the crowd, and reference alignments) and compute precision and recall values.

The functionality offered by CrowdMap could be easily integrated into existing environments for ontology alignment, such as the PROMPT Protégé plug-in [21] or even used to complement tools that perform data interlinking, such as Silk [22] and Google Refine with curated information about schema-level alignments.

3.4 Microtask User Interface Design

In CrowdFlower, the user interface that a worker sees has three main parts: (i) the title and instructions explaining the purpose of the microtask; (ii) the problem statement, which in our case is the information about the elements (e.g.,classes) to be compared; and (iii) the form that workers must fill out to submit their responses. CrowdMap defines two types of microtasks for which we generate different interfaces: (i) validation microtasks and (ii) identification microtasks. A validation microtask presents workers with a complete mapping (e.g., two classes and the relationship that connects them) and asks them to specify whether they agree with the relationship that they see.


Concept A: Misc
Definition (English): Use this type when nothing else fits.

Misc *is a kind of:* Reference

Other elements that are of kind Reference: 'Academic' 'Informal' 'MotionPicture'

Concept B: Misc.
Definition (English): Use this type when nothing else fits.

Misc. *is a kind of:* REFERENCE

Other elements that are of kind REFERENCE: 'Book' 'Academic' 'Motion_picture'

Is Concept A the same as Concept B? (required)
○ yes
○ no
Please select only one of the answers

Select the name of Concept A (required)
○ Misc
○ Misc.
Please select only one of the answers

How many distinct words are in the name of Concept A? (required)

Please write the number in the text box

Fig. 2. User interface of a validation microtask. CROWDMAP shows the worker two elements to be aligned and asks whether they are related to each other with a particular relationship.

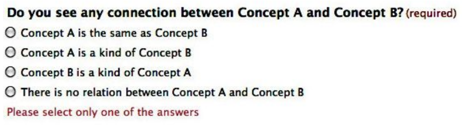

Do you see any connection between Concept A and Concept B? (required)
○ Concept A is the same as Concept B
○ Concept A is a kind of Concept B
○ Concept B is a kind of Concept A
○ There is no relation between Concept A and Concept B
Please select only one of the answers

Fig. 3. User interface of an identification microtask where CROWDMAP shows the worker two elements to be aligned and asks to identify the relationship between them. The relationship in this case can be that both are the same, one is more specific than the other, or the two are not the same.

An identification microtask asks for workers to identify a particular relationship between the source and the target classes. Figure 2 shows an example of a validation microtask. The first part is the problem statement; the second part is the form. The microtask includes all contextual information available for both classes (labels, definitions, superclass, siblings, subclasses and instances). The first element in the form asks the user whether or not the concepts are similar. The form also includes two more elements as verification questions that help in filtering spam, similarly to the approach by Kittur and colleagues [16]. We use a different input form for identification microtasks. Figure 3 shows the first field of three sample questions within an identification microtask. CROWDMAP can create identification microtasks showing either a complete version of the form (relationships =, <=, >=, none), or a short version (=, not =). Anti-spam mechanisms are the same as for validation microtasks, illustrated in Figure 2.

In order to reduce response bias, CROWDMAP creates only half of the HITs using the interface in Figures 2 and 3. In the other half, CROWDMAP presents the possible answers in the opposite order, and focus the verification question on the other class in the pair to be matched. This technique, which we apply independently from the type of microtask, makes the evaluation of workers stricter, allowing us to identify and block spam more efficiently. The verification questions that we used to identify and avoid spam play a special role in these checkpoint-like questions; the response of a worker to a golden unit is evaluated positively only if all three fields of the input form have a correct response.

4 Evaluation

In order to perform our analysis, we conducted several studies to test both the feasibility of overall approach and specific characteristics of the design of crowdsourced ontology alignment that improve its effectiveness. We used the ontologies and the reference alignments from the Ontology Alignment Evaluation Initiative (OAEI) as golden standard to assess the accuracy of the crowd-computed results.

4.1 Ontologies and Alignment Data

We have conducted three sets of experiments in order to address the research questions from Section 1 (Table 4.1).

In our first experiment, CARTP, candidate mappings included all possible pairs of mappings between two input ontologies (a Cartesian product of the sets of classes). While such an approach does not scale in practice, it provides the baseline on the best possible performance (recall in particular) of crowdsourced alignment. The OAEI ontologies that we use for the CARTP experiment are two ontologies that cover the BibTex data, one from MIT and one from INRIA (ontologies 301 and 304 from the OAEI set). For each pair of classes we provide the user with contextual information that is relevant to the corresponding elements and compare the results against the reference alignments provided by the OAEI.

The second type of microtasks, which we call IMP, uses only those class pairs that were created by a given ontology alignment tool as a set of candidate mappings. This experiment simulates a typical CROWDMAP workflow (Figure 1). We used the output of the AROMA tool as our input alignment. AROMA is one of the algorithms from OAEI that presented a good performance in 2011. Again, we ran the experiment using ontologies 301 to 304 just as in the CARTP and included full context-specific descriptions of the two elements to be matched. Note that we obtained the results for the IMP setup by using the CARTP data since we already had the judgements for all the pairs of terms from the two ontologies that we used in both experiments.

The third set of microtasks, which we call 100R50P, includes several ontology pairs and allows us to compare the CROWDMAP performance in different settings. The sets of candidate mappings in the 100R50P experiments simulate input originating from a tool with 100% recall and 50% precision. We create the set of class pairs where 50% of the mappings are correct and 50% are incorrect. We take the correct mappings from ontology alignment reference data. Incorrect mappings consist of false negatives (generated by an algorithm), as well as randomly generated mappings. If there is no algorithm to generate candidate alignments, we generate all the incorrect mappings by selecting pairs of classes randomly.

We use the *Conference ontologies* from the OAEI set. The ontologies in this set represent knowledge about conferences and were produced by different organizations. Some of the selected ontologies are based on actual tools for conferences (Cmt and ConfOf), and others are based on either personal experiences (Ekaw) or Web pages of conferences (Sigkdd). We took a pair of ontologies from this set, choosing the Argmaker algorithm results as the alignments performed by the automatic tool.

Table 1. Summary of the experiments

	CartP	Imp	100R50P
Ontologies	301-304	301-304	Edas-Iasted, Ekaw-Iasted, Cmt-Ekaw, ConfOf-Ekaw
Input alignment	Cartesian product	Output of the AROMA algorithm	50% correct mappings (all mappings from the reference alignment), 50% incorrect mappings, output of AgrMaker
Research question	R1	R2	R2, R3

The ontologies in the OAEI *Oriented matching* set cover the domain of academia and the reference alignment includes complex relationships, such as broader than and narrower than. We took the same pair from this set that we used in the CartP experiment (301 to 304).

Table 4.1 summarizes the three experiments.

4.2 CrowdFlower and MTurk Setup

Both CrowdFlower and MTurk allow requesters to configure their microtask projects according to a number of different parameters. In our experiments, we clustered 7 different alignment questions (or units in CrowdFlower parlance) into one HIT. This step facilitates worker assignment and resource optimization (see Section 3.1). One of these questions was a golden unit (see Section 3) where we knew the answer in advance. We could use it to assess the performance of workers, to deal with spammers, and to validate the final results. For each experiment we selected golden units from a set of 50. Each HIT includes two verification questions, which apply to both golden and real units, as a means to reduce spam (see Section 3.4).

Redundant answers to the same question are a useful way to evaluate the feasibility of the overall approach—can users actually agree on the answer?— and to (automatically) identify correct answers. We requested 3 workers for those questions that asked them whether a given correspondence holds or not. These values are based on best practices in crowdsourcing literature [1].

It is common for microtask platforms to organize HITs in batches. In our case, each batch contained at most 50 HITs, each with 7 mapping units. This value is an empirical one used in similar experiments on MTurk [16], which balances resource pooling and the time required to complete a full batch. Several workers verified each alignment, not only to receive the minimal number of answers required for majority voting, but also because we wanted to change the order of the allowed answer choices to avoid spammers. We calculated the number of golden units as the number of HITs in each group, and adjusted the number of mappings to show in each set of alignment questions, in cases where it was needed by the CrowdFlower internal restrictions. CrowdFlower requires that a worker answers 4 golden units correctly before she becomes a trusted workers. We reduced this number to 2 since we observed that workers were submitting fewer than 4 correct answers to golden units, even with correct mapping results.

For most experiments we paid $0.01 for each HIT; for the CartP scenario we raised the reward to $0.04 to compensate for the larger scale of the experiment and to study the trade-offs between time to completion and costs. CrowdFlower publishes jobs on

Table 2. Precision and recall for the crowdsourcing results

	CARTP 301-304	100R50P Edas-Iasted	100R50P Ekaw-Iasted	100R50P Cmt-Ekaw	100R50P ConfOf-Ekaw	IMP 301-304
Precision	0.53	0.8	1.0	1.0	0.93	0.73
Recall	1.0	0.42	0.8	0.75	0.65	1.0

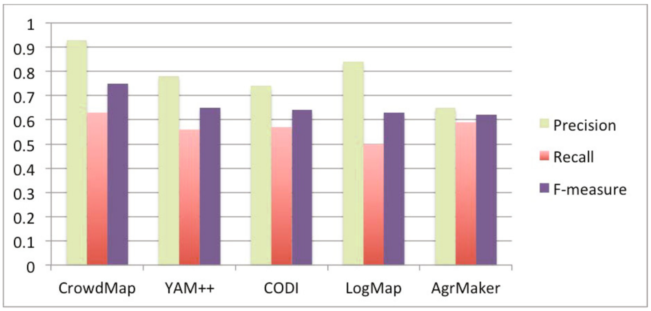


Fig. 4. The average precision, recall, and F-measure of CROWDMAP and the top performers on the conference set for OAEI 2011 (http://oaei.ontologymatching.org/2011/results/conference/index.html)

the platform for 7 days by default and the deadline is extended until jobs are completed. For most of the experiments we needed between 7 and 10 days, which is possibly also a consequence of the fact that we published several similar jobs within a relatively short period of time. The higher-rewarded experiments required less than a day to finalize, which was significantly faster than other trials we ran on the same data and $0.01 per HIT.

4.3 Results

Table 2 shows the precision and recall in our experiments. We use the PRecEvaluator available in the Alignment API to calculate these values.

The results show very high precision for the conference alignments. Figure 4 compares the performance of CROWDMAP on the conference set with the 4 top performers in OAEI 2011. The chart shows the average precision, recall, and F-measure. Note that CROWDMAP significantly outperforms the other algorithms, with the F-measure of 0.77. It is important to note, however, that for the conference set, CROWDMAP does not start with a cartesian product of all possible pairs. It needs to filter only a set of mappings that have 50% correct mappings and 50% wrong mappings. However, the crowd improved the precision considerably from that 50%.

For the CARTP alignment, the workers have found all the mappings from the reference alignment, achieving a remarkable 1.0 recall. The precision, however, has suffered. We address this issue in our discussion in Section 5.

5 Analysis and Lessons Learned

The results of the experiments lead to the following conclusions

R1 From the result achieved in the CARTP experiment we can conclude that our approach is feasible. Given the full set of potential correspondences between pairs of classes, the crowd was able to provide meaningful answers that could be used in the alignment process.

R2 If we compare the results of 100R50P with the performance of the AgrMaker algorithm which we used as a baseline on the conference alignments (precision: 0.65 and recall: 0.59), we notice that CROWDMAP can improve both the precision and the recall of the original algorithm. This finding is supported by the outcomes of the IMP experiment, by comparison with performance of the AROMA tool on the same alignment (precision: 0.62 and recall: 1.0).

R3 Workers were capable of submitting correct responses with both validation and identification microtasks.

One unexpected observation from Table 2 is the effect of the number of mappings that we present on precision. The precision is very high for all the tasks in the 100R50P set, where we showed only a limited number of pairs of mappings. In the CARTP experiments, the workers had access to the cartesian products of all the class pairs, and the precision dropped significantly. Because we used the CARTP results to simulate the IMP experiment, the precision there suffered as well. Our hypothesis for this low precision is that the large task might attract more spammers or more workers just try to get through the task quickly. However, in future work, we plan to design experiments to test this hypothesis.

However, if we look at results from the different experiments together, we can see a potential for a two-step process that might be very efficient. The workers can achieve perfect (or close to perfect) recall when given a large set of candidate pairs, many of which are not mappings. They achieve high precision on a set that has fewer wrong mappings but all correct ones. Thus, we can use a setting such as CARTP (extremely low precision, perfect recall) to get a set that is close to 100R50P (50% precision, 100% recall). Indeed, CARTP produced a mapping set that was extremely close to an 100R50P set. This approach would create a two-step CROWDMAP algorithm: first stage uses CARTP (or its approximation, by taking all the very low confidence mappings from an automatic tool). Then we can use the results of this first stage as an input to another run of CROWDMAP which will improve the precision. Note that this approach is similar to the Find-Fix-Verify crowd programming pattern in Soylent [4].

We carried out the experiments over a period of five weeks, whereas half of this time was dedicated to the tuning of the configuration parameters of the crowdsourcing platform and the testing of different variants of the interfaces (see Section 3.4). In its current, optimized version, we estimate that CROWDMAP could produce accurate alignments between pairs of ontologies within a relatively short period of time (around one week for several hundreds of HITs and corresponding alignments). The total costs of the experiments were around \$50, which is not comparable to alternative approaches oriented at knowledge engineers or domain experts, with or without the involvement of automatic algorithms.

Before running the experiments that we reported, we tested the prototype with small pilots. The pilots allowed us to fine-tune the user interface and to develop methods to minimize spam. When we initially did not use golden units or verification questions, we received a huge amount of spam. While we collected the required responses in a few hours, most of them appeared to be very low quality ones. Over several iterations, each of which reduced the number of spam, we came to the following strategies. First, we use golden units to block invalid answers. Second, we use verification questions that force the user to type a name of the concept. Finally, CrowdFlower allows requester to exclude specific countries that have workers who tend produce the majority of spam answers. Including developing countries such as India was another strategy that helped reduce spam significantly.

The wording and structure in the user interface also influenced the results. We experimented with different types of verification questions and phrasings thereof. We wanted to define additional questions that were trivial to answer, yet, required the user to process cognitively the information on the form. We also needed verification questions that would get different answers from one pair of terms to the next, so that workers could not cut and paste. In the experiments that we report here, we used both the names of the classes to be compared, as well as other features such as the number of words in the class names as basis for such verification questions. For one type of verification question asking for the name of one the classes to be matched, we eventually decided in favor of a radio button rather than a free-text field, as in the latter case many workers simply typed in the default name 'Concept A' mentioned in the question. References to the "first" or "second" class in the matching pair also turned out to confuse users. In the case of a second verification question, which asks about the number of distinct words displayed, a simple validator encouraged workers using positive integers (e.g., "1") instead of text (e.g. "one"), and thus avoiding correct responses to be evaluated negatively. Changing the wording of equivalence-alignment questions from "Concept A is similar to Concept B" to "Concept A is the same as Concept B" lead to a better understanding of the task by the workers and to better results. Finally, we verified how important ontology documentation is, since CROWDMAP relies on the quality of labels and definitions.

Another observation that we made is related to the number of related microtasks (or groups of questions) published at the same time; in this case the time to completion increased, probably due to the fact that the same workers typically take the opportunity to solve a series of similar tasks. The results that we have obtained largely depend on the data set used for the evaluation. It is worthwhile mentioning that, there have been cases in which the crowd identified mappings that were correct in our opinion (such as $Person - Person$), but were not present in the reference alignment. This means that these mappings did not count for the recall and precision values. We also analyzed the mappings that the crowd missed from the reference alignment, and we must say that there were cases that were not clear for us either. For example, mappings such as $WelcomeTalk - Welcome\_address$, or $SocialEvent - Social\_program$, or $Attendee - Delegate$ (from test $Edas - Iasted$) are ambiguous.

Most work on using crowdsourcing for computational tasks rely on MTurk as a platform. Our experiences with CrowdFlower showed that this platform represents a real alternative to directly accessing the MTurk crowd, in particular due to the additional features they offer with respect to quality assurance. However, it is worthwhile mentioning that while it is possible to use MTurk via CrowdFlower, the latter does not support the full range of services of the former; for instance, it is not possible to update the number of answers required for a question during the execution of a task.

6 Conclusions and Future Work

This paper makes several contributions to the state of the art in ontology alignment. First, we present a workflow model for crowdsourcing ontology mappings and describe the implemented solution that uses CrowdFlower. Second, we perform a feasibility study for the use of crowdsourcing to perform ontology mapping. Third, we provide an analysis of the characteristics of crowdsourced ontology mappings for different ontologies, mapping relationships, and settings. Our first prototype of CROWDMAP has proven that the crowdsourcing approach to ontology alignment is feasible, and can augment automatic tools in a cost-efficient, fast, and scalable manner.

Future work will focus on executing new experiments to analyze further research questions. For example, we would like to discover which contextual aspects are the most useful to improve accuracy, and whether we could use agreement among workers to determine the certainty of mappings. We expect to create a set of instances for each ontology used in the experiments, so that workers can see up to 5 instances as part as the context of the elements to be aligned. We will perform more experiments to test whether accuracy is reduced in cases where the domain of the ontologies requires specific knowledge (e.g., biomedical ontologies). Finally, after completing the extensive set of experiments, we believe that we can improve the worker performance by fine-tuning the question wording even better (e.g., substituting the class names directly into the options for selection in the mapping questions).

We plan an extension of the implemented prototype of CROWDMAP to enable crowdsourced mappings between ontology properties and axioms. With respect to the actual workflow, we will look into more sophisticated means to combine the results of human and algorithmic computations, by following, for instance, a Bayes analysis approach (cf. [15]). Along the same lines, we also intend to apply filtering techniques to optimize the number of questions that are issued to the crowd to improve scalability and costs. Such filtering is an essential pre-requisite for the application of CROWDMAP to related fields such as data interlinking, which has orders or magnitude more data and possible a larger degree of noisy data than the scenario that we studied in this paper.

Acknowledgements. We would like to thank the self-service team of CrowdFlower, for their technical support on the CrowdFlower API.

References

1. Ipeirotis, P., Provost, F., Wang, J.: Quality management on Amazon Mechanical Turk. In: Proceedings of the ACM SIGKDD Workshop on Human Computation, pp. 64–67 (2010)
2. Kulkarni, A., Can, M., Hartmann, B.: Turkomatic: automatic recursive task and workflow design for Mechanical Turk. In: Human Factors in Computing Systems, CHI (2011)

3. Little, G., Chilton, L., Goldman, M., Miller, R.: TurKit: tools for iterative tasks on mechanical Turk. In: Proceedings of the ACM SIGKDD Workshop on Human Computation, pp. 29–30 (2009)
4. Bernstein, M., Little, G., Miller, R., Hartmann, B., Ackerman, M., Karger, D., Crowell, D., Panovich, K.: Soylent: a word processor with a crowd inside. In: Proceedings of the 23nd Annual ACM Symposium on User Interface Software and Technology, pp. 313–322 (2010)
5. Shi, F., Li, J., Tang, J., Xie, G., Li, H.: Actively Learning Ontology Matching via User Interaction. In: Bernstein, A., Karger, D.R., Heath, T., Feigenbaum, L., Maynard, D., Motta, E., Thirunarayan, K. (eds.) ISWC 2009. LNCS, vol. 5823, pp. 585–600. Springer, Heidelberg (2009)
6. Falconer, S.M., Storey, M.-A.: A Cognitive Support Framework for Ontology Mapping. 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. 114–127. Springer, Heidelberg (2007)
7. McCann, R., Shen, W., Doan, A.: Matching Schemas in Online Communities: A Web 2.0 Approach. In: 18th International Conference on Data Engineering (ICDE), pp. 110–119 (2008)
8. Hausenblas, M., Troncy, R., Raimond, Y., Bürger, T.: Interlinking multimedia: How to apply linked data principles to multimedia fragments. In: WWW 2009 Workshop: Linked Data on the Web (2009)
9. Noy, N.F., Griffith, N., Musen, M.A.: Collecting Community-Based Mappings in an Ontology Repository. In: Sheth, A.P., Staab, S., Dean, M., Paolucci, M., Maynard, D., Finin, T., Thirunarayan, K. (eds.) ISWC 2008. LNCS, vol. 5318, pp. 371–386. Springer, Heidelberg (2008)
10. Zhdanova, A., Shvaiko, P.: Community-driven ontology matching. Technical Report DIT-06-028, Ingegneria e Scienza dell'Informazione, University of Trento (2006)
11. 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)
12. Thaler, S., Siorpaes, K., Simperl, E.: SpotTheLink: A Game for Ontology Alignment. In: Proceedings of the 6th Conference for Professional Knowledge Management (2011)
13. Thaler, S., Siorpaes, K., Mear, D., Simperl, E., Goodman, C.: SeaFish: A Game for Collaborative and Visual Image Annotation and Interlinking. 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. 466–470. Springer, Heidelberg (2011)
14. Markotschi, T., Völker, J.: GuessWhat?! - Human Intelligence for Mining Linked Data. In: Proceedings of the Workshop on Knowledge Injection into and Extraction from Linked Data at EKAW (2010)
15. Demartini, G., Difallah, D.E., Cudré-Mauroux, P.: ZenCrowd: leveraging probabilistic reasoning and crowdsourcing techniques for large-scale entity linking. In: Proceedings of the 21st World Wide Web Conference, WWW 2012, pp. 469–478 (2012)
16. Kittur, A., Chi, E., Suh, B.: Crowdsourcing user studies with Mechanical Turk. In: Proc. 26th Annual SIGCHI Conf. on Human Factors in Computing Systems, pp. 453–456 (2008)
17. Franklin, M., Kossmann, D., Kraska, T., Ramesh, S., Xin, R.: CrowdDB: answering queries with crowdsourcing. In: Proceedings of the 2011 International Conference on Management of Data, SIGMOD 2011, pp. 61–72 (2011)
18. Bernstein, M., Karger, D., Miller, R., Brandt, J.: Analytic Methods for Optimizing Realtime Crowdsourcing. CoRR abs/1204.2995 (2012)

19. Oleson, D., Sorokin, A., Laughlin, G., Hester, V., Le, J., Biewald, L.: Programmatic gold: targeted and scalable quality assurance in crowdsourcing. In: AAAI Workhop on Human Computation (2011)
20. David, J., Euzenat, J., Scharffe, F.: The Alignment API 4.0. Semantic Web Journal 2(1), 3–10 (2011)
21. Noy, N.F., Musen, M.A.: The PROMPT suite: Interactive tools for ontology merging and mapping. International Journal of Human-Computer Studies 59(6), 983–1024 (2003)
22. 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)

Domain-Aware Ontology Matching

Kristian Slabbekoorn[1], Laura Hollink[2,3], and Geert-Jan Houben[2]

[1] Department of Computer Science, Tokyo Institute of Technology, Japan
[2] Web Information Systems Group, Delft University of Technology, The Netherlands
[3] Knowledge Representation and Reasoning Group, VU Amsterdam, The Netherlands

Abstract. The inherent heterogeneity of datasets on the Semantic Web has created a need to interlink them, and several tools have emerged that automate this task. In this paper we are interested in what happens if we enrich these matching tools with knowledge of the domain of the ontologies. We explore how to express the notion of a domain in terms usable for an ontology matching tool, and we examine various methods to decide what constitutes the domain of a given dataset. We show how we can use this in a matching tool, and study the effect of domain knowledge on the quality of the alignment.

We perform evaluations for two scenarios: Last.fm artists and UMLS medical terms. To quantify the added value of domain knowledge, we compare our domain-aware matching approach to a standard approach based on a manually created reference alignment. The results indicate that the proposed domain-aware approach indeed outperforms the standard approach, with a large effect on ambiguous concepts but a much smaller effect on unambiguous concepts.

1 Introduction

The rise of the Semantic Web and Linked Open Data has led to a large number of heterogeneous datasets and ontologies published on the Web [2]. This kind of heterogeneity has created a need to interlink these datasets, i.e. to make explicit that two resources of different datasets represent the same concept. However, the discovery of such correspondences between entities of different ontologies or datasets is not trivial – it is in fact one of the major challenges of the Semantic Web [1].

A large number of ontology matching tools and methods have emerged in recent years [4]. In this paper we are interested in what happens if we take these existing tools and enrich them with knowledge of the domain of a given domain-specific ontology. This is especially relevant in situations where we want to match such an ontology to a more general ontology; a scenario that is often seen on the Linked Data Web. If our source dataset is, for example, a medical vocabulary, it would help us a lot to know which concepts in the target dataset are within the relevant domain of medicine, as disambiguation of concepts is not trivial. In this paper we examine various methods to decide what constitutes the domain of a given dataset. We explore how to express the notion of a domain in terms usable for an ontology matching tool and we study the effect of domain knowledge on the quality of the alignment. The main contribution is the development and evaluation of a domain-aware ontology matching approach geared at interlinking domain-specific ontologies with cross-domain ontologies.

We present an approach that, in a fully unsupervised way, discovers the domain of a domain-specific source ontology expressed in terms of the schema information of the (cross-domain) target ontology. This ensures that matching can be performed by restricting matches to entities within this domain. The approach consists of five phases: (1) a *high-confidence matching phase* in which we determine a small set of mappings between source and target ontology for which we are almost sure they are correct; (2) a *class/category collection phase* in which we collect the schema information of the mapped instances of the target ontology; (3) a *domain pruning phase* in which we translate the domain knowledge into a filter; (4) a *domain optimization phase* in which we optimize this filter; and finally (5) a *domain-aware matching phase* where the actual matching is performed – the domain filter is used in an ontology matching tool to ensure that concepts are only matched if they are within the relevant domain.

We evaluate our approach in two scenarios: we match (1) artists mined from Last.fm in the EventMedia dataset of cultural events[1] and (2) a part of the medical vocabulary UMLS to DBpedia. We have chosen DBpedia because this is the de facto cross-domain centerpiece of the Linked Open Data cloud. This makes it an attractive choice as it contains many concepts (over 3.64 million on September 2011) across various domains, presenting us with the real problems of ambiguous concepts and thus irrelevant candidate matches. By focusing on a single target ontology, we aim to achieve a matching approach that yields high accuracy for the generated links. In addition, it allows us to optimally make use of the schema information that is specific to the target ontology.

In both scenarios, we experiment with different methods to decide what constitutes the domain of a dataset. We use each method in combination with DBpedia Spotlight, a dedicated DBpedia matching tool, to create an alignment. To quantify the effect of domain knowledge in general, and each of the domain-derivation methods specifically, we compare each alignment to a manually created reference alignment. An early version of our approach, with a more limited domain representation and without a domain optimization phase, was evaluated on one of these scenarios and has been published previously [20].

The remainder of this paper is organized as follows. We start by giving a description of the task we are going to perform in section 2. Next, we explain the domain-aware approach, in section 3. Section 4 contains the evaluation of the approach and a discussion of the results. In section 5 we present a discussion of related work, expanding on works dealing with similar approaches to ontology matching. The final section concludes the paper and describes possible future work.

2 Task Description

Domain Representation in DBpedia. As mentioned, we have chosen DBpedia as the target ontology of our approach, which allows us to use the schema information of DBpedia in the representation of a domain filter. This goes further than built in schema constructs in standard Semantic Web languages: to represent a domain we use *DBpedia classes*, *Wikipedia categories*, *YAGO classes* and *Freebase classes*. The first three types are already available in DBpedia; Freebase classes are derived through Freebase

[1] `http://thedatahub.org/dataset/event-media`

interlinks. Therefore, what we essentially want to find out is how the domain of the source ontology can be mapped effectively to a set of of classes and categories in DBpedia.

Given the full union set of all classes and categories C, a *domain* D can be defined as a subset of C, i.e. $D \subseteq C$. Once we have derived a target domain, we can use it as a filter for the traditional matching process done by (complex) string comparison. The reason we derive a set of classes/categories rather than a single class is that there is usually no perfect mapping to a single target class. Also, the classes contained in DBpedia are either user-generated, or derived from a user-generated base. Therefore, since not all resources have proper classes or categories assigned to them, some redundancy in the domain encapsulation helps to deal with this messiness of data (figure 1).

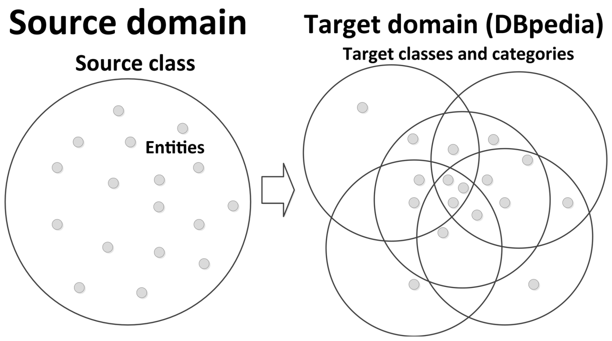


Fig. 1. Mapping from source to target domain

Source Datasets and Quality Measures. We evaluate the approach with two domain-specific ontologies: Last.fm artists and UMLS medical concepts. The Last.fm Artist dataset consists of roughly 50,000 entities. For the UMLS concepts we consider two separate classes of terms: pathologic terms and physiologic terms, which consist of roughly 70,000 resp. 55,000 terms. To illustrate the approach more clearly, we take the Last.fm Artist dataset as running example. Note that these source datasets are used for evaluation purposes only; our approach is independent of the source dataset.

Manual reference alignments R are created for all three datasets. For Last.fm Artists we manually determine links for 1000 randomly selected artists; for each type of UMLS terms we create links for 500 random entities. The links are determined by one person (the first author) – we feel that this suffices, as the reference alignments are used to compare variations of our method (and a baseline), not to give an authoritative performance score to an alignment method. We measure the quality of alignments in terms of F-measure, which is the harmonic mean of recall and precision. Our task is to produce alignments with a high F-score.

Baseline Matching Tool: DBpedia Spotlight. As a starting point for the actual matching task we take DBpedia Spotlight [14], a powerful, off-the-shelf tool for matching textual resources to DBpedia. DBpedia Spotlight has been shown to be able to compete with established annotation systems while remaining largely configurable [14]. Its configurability allows us to include various forms of domain knowledge, and test the effect on the resulting matches.

Spotlight works by first finding *surface forms* in the text that could be mentions of DBpedia resources (the “spotting” function), then disambiguating these surface forms to link to the right DBpedia resources based on context similarity measures (the “disambiguation” function). Its results can be directed towards high precision or high recall by setting two parameters: a “support” threshold for minimum popularity of the associated Wikipedia page (i.e. the number of inlinks to this page from other Wikipedia pages) and a “confidence” threshold for minimum similarity between source text and context associated with DBpedia surface forms.

For our work we do not employ the full range of Spotlight’s features – the tool mostly specializes in the annotation of free text, while we deal with a very specific case of text annotation, namely that of entity labels. In the majority of cases it suffices to simply try to disambiguate this full label to find a corresponding DBpedia resource.

3 The Domain-Aware Matching Approach

Figure 2 shows a high-level schematic of the pipeline employed. As input we take an ontology or dataset, and the label(s) and context of each entity. Given the example of the Last.fm dataset, we want to match artists, so we give as input the `foaf:Agent` class of the EventMedia RDF dataset, and as label and context properties `rdfs:label` and `dc:description` respectively.

The approach is then divided into five phases. In the *high-confidence matching phase*, we collect an initial sample of links of which we can be fairly sure they are correct. For each DBpedia resource in this sample of links, classes and categories are collected in the *class/category collection phase*. This collection is pruned to a processable size in the *domain pruning phase*, then optimized to form an adequate domain filter in the *domain optimization phase*. Lastly, the domain-restricted matching is performed in the *domain-aware matching phase*. Each step will be explained in detail in the following sections.

3.1 High-Confidence Matching Phase

Initially, we assume to have no knowledge about our input dataset. The selection of the domain filter is bootstrapped by first attempting to match instances of the source dataset to DBpedia resources without any knowledge of the domain, to obtain an initial linkset from which we want to derive domain information. Ideally, we want to maximize precision for these links while still obtaining a sufficiently large sample to make meaningful generalizations for the full dataset. This is accomplished by applying DBpedia Spotlight

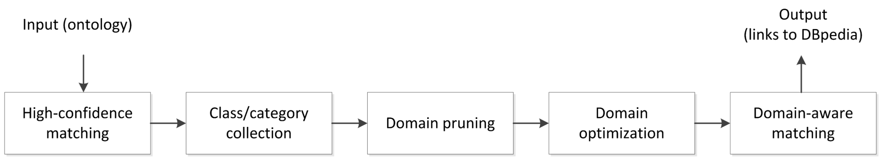


Fig. 2. High-level pipeline view for the domain-aware approach

without any domain restriction and with parameters set towards high precision, thereby relying solely on the lexicographical similarity of contexts between source and target entities for disambiguation (entity labels and descriptions are compared to the textual content of candidate DBpedia resources). Entities to match are selected at random from the source dataset. In this way, we can generate an initial set of high-confidence links from our data to DBpedia resources. This set of links is used for two separate purposes: (1) the collection of classes and categories for the derivation of a domain; and (2) the optimization of the target domain filter (as discussed in section 3.4).

For the first goal, we want a linkset that is large enough to make accurate predictions about the domain. For the second goal, we want a set of links that resembles a reference alignment, consisting of *positive matches* and *negative matches*. We call the set of high-confidence matches the *high-confidence linkset*.

Definition 1 (Positive match). *A mapping from a source entity to a target DBpedia resource. A single source entity may have multiple positive matches to (distinct) DBpedia resources.*

Definition 2 (Negative match). *A null-mapping for a source entity, i.e. this entity is determined to have no corresponding DBpedia resource.*

Definition 3 (High-confidence linkset). *A* high-confidence linkset R' *is the set of positive and negative matches obtained by bootstrapping the source dataset.*

For the purposes of this experiment, the high-confidence linkset was made to be disjoint with the reference alignment R. The variable name R' was chosen as such because the high-confidence linkset can be regarded as a *virtual* version of a real reference alignment R. For evaluation purposes, we fix the number of matches in R' to 1000. We have experimented with high-confidence linksets containing only positive matches, and linksets containing both positive and negative matches in various ratios. We found that the best results were obtained with a high-confidence linkset containing positive and negative matches in a 500/500 ratio ($R'_{500/500}$). For the sake of brevity, we omit these experiments from this paper and take $R'_{500/500}$ as high-confidence linkset type.

The best value to choose for “confidence,” the threshold parameter for context similarity in Spotlight, is somewhat dependent on the dataset. The positive and negative matches are gathered with two *separate* confidence thresholds: to be maximally confident in positive matches, the threshold should be high, i.e. high context similarity and/or low ambiguity are demanded; to be maximally confident in negative matches, the threshold should be low, i.e. low context similarity and/or high ambiguity are demanded. We aim for the strictest thresholds where enough matches are still obtained.

The resulting high-confidence linkset allows us to derive and optimize a domain filter without the use of a manually created reference alignment.

3.2 Class/category Collection Phase

From the resulting DBpedia resources in the high-confidence linkset we gather the associated classes and categories so that we can later use these as a domain knowledge filter for a matching tool. We gather *DBpedia classes*, *Wikipedia categories*, *YAGO classes* and *Freebase classes*. The first three types are already available in DBpedia; Freebase classes are derived through Freebase links in DBpedia. We use the following collection of classes/categories as a basis for the derivation of an effective domain filter:

1. *DBpedia classes.* These are derived from the manually assigned infobox types on Wikipedia pages. The DBpedia classes are ordered in a shallow, strict hierarchy (i.e. each class has no more than one super-class), so we use the transitivity of the subclass hierarchy and include not only classes directly associated to the resource, but also all super-classes up to the root of the hierarchy.
2. *Wikipedia categories.* These are the categories manually assigned to Wikipedia pages. They are ordered in a broad and non-strict hierarchy (i.e. categories can have multiple super-categories). Therefore we gather only up to 4 ancestors of each, as due to the size and messy structure of the category hierarchy the set would quickly become too large and broad to be useful.
3. *YAGO classes.* These are automatically derived from the Wikipedia category system using WordNet [15][23]. YAGO classes are ordered in a deep, near-strict hierarchy (a select few classes have multiple super-classes), so we gather all super-classes up to the root of the hierarchy.
4. *Freebase classes.* These are taken from the Freebase concept associated with a DBpedia resource (if available). Freebase classes are ordered in a shallow, strict hierarchy of only two levels with no single root node, so all we gather for each concept is the class and its super-class.

3.3 Domain Pruning Phase

We prune the collection of classes and categories in order to get a tighter encapsulation for the domain. Classes that only occur once or twice over the entire class/category set are typically better left out, as these are likely to be outliers to the domain of interest. Even if not, they tend to be too narrow to play any role. In addition, we need to discard classes that are too broad and cover far more entities than what we want it to do. This is especially necessary as we collect DBpedia and YAGO classes up to the root of the hierarchy, thus including the root classes Thing and Entity.

We develop two *pruning strategies* that can be applied to somewhat limit the initial domain, all the while taking care not to discard potentially relevant domain information.

Low-occurrence domain pruning (LDP). We can get rid of classes/categories that occur too little as follows. We filter out all DBpedia and YAGO classes that occur less than $r_d\%$ resp. less than $r_y\%$ of the total number of classes found. A percentage is required since we do not know beforehand exactly how many of the top t most occurring classes are too generic, as this depends on the domain of the source dataset. For categories, we do not necessarily need to filter by a percentage since we do not gather super-categories up to the root. Since we only collect them within a specified semantic distance, the very categories with the most occurrences should be a relatively accurate representation of our domain. Therefore we simply select the top t_c categories that occur the most. For Freebase classes, there are only two levels in the hierarchy, hence this effect does not occur here either: we select simply the top t_f classes.

High-generality domain pruning (HDP). A second strategy is needed to eliminate excessive redundancy in the list of classes and categories, i.e. to avoid including a super-class plus all of its subclasses. We address this by filtering out all super-classes where the sum of the numbers of occurrence of their *direct* sub-classes is greater than $p\%$ of the number of occurrence of the super-class. In other words, for every class (or category) *Super*, if

$$\sum_{i=1}^{n} occs(Sub_i) > \frac{p}{100} occs(Super), \quad (1)$$

where n is the number of direct subclasses Sub_i of *Super*, then *Super* is removed from the collection. This procedure is dependent on the values chosen for r_d, r_y, t_c and t_f in the LDP step.

See figure 3 for an illustrated example of pruning Last.fm artists with $r_y = 1\%$ (cutoff of 14 occurrences in this case) and $p = 80\%$ for the YAGO hierarchy. Class occurrence numbers are shown in parentheses. The percentages are the proportions of direct sub-class occurrences to this class' occurrences. Classes grayed out were discarded in the LDP step; classes marked red are discarded in the HDP step. Classes marked green represent the final domain. As can be seen, most generic and unrelated classes are successfully filtered out. However, some broad classes, such as "LivingPeople," still remain, which suggests that we need extra measures in order to further refine the domain encapsulation.

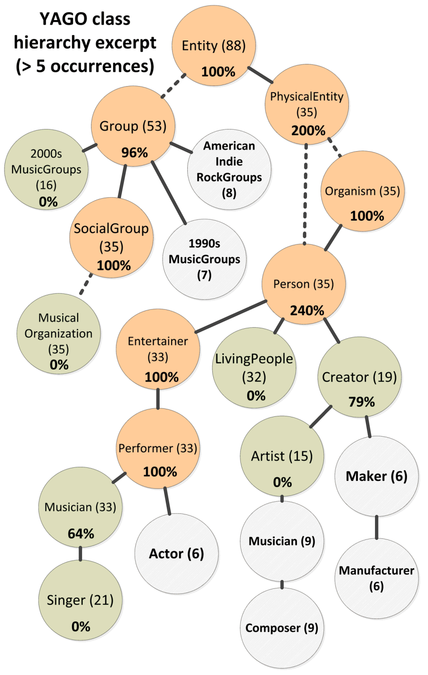


Fig. 3. Top part of the YAGO hierarchy after pruning with $r_y = 1\%$ and $p = 80\%$

3.4 Domain Optimization Phase

After pruning classes/categories that are too general or too specific, we further optimize the set of classes/categories in the domain filter by testing which set gives the highest alignment performance. We apply and compare three methods. The first is a metaheuristic-based approach [13], i.e. a generic optimization method that is known to be able to discover (near-)optimal solutions independent of the nature of the optimization problem itself. For this we will apply a genetic algorithm to our problem. The second and third are problem-specific heuristic-based approaches that exploit the structure of our specific problem domain.

Each optimization approach needs a way to evaluate the candidate solutions. As said, we aim to be independent of a manually created reference alignment and instead use the automatically created high-confidence linkset. This approach is only sensible, however, under the assumption that there exists a correlation between performance on the automatically generated high-confidence linkset R' and performance on the manually constructed reference alignment R. Before we go into the details of the three optimization approaches, we first show that this assumption is valid.

Testing the Validity of the Bootstrap Linkset. We test various configurations of a domain-aware matcher by comparing the alignments that it creates to the manually created reference alignment and to the high-confidence linkset. Performance on the reference alignment is measured with the F-score. Performance on the high-confidence linkset is measured in the same way, but we call this measure the "virtual" F'*-score* to distinguish it from the "real" F-score.

We evaluate by comparing F'-scores obtained using R' and F-scores obtained using R. We generate 36 different domain filters by first applying the high-confidence matching and domain pruning phases of sections 3.1 and 3.3 with varying settings, and then randomly excluding half of the classes/categories. Each of these 36 domain filters is used in combination with DBpedia Spotlight to produce two alignments: one on the concepts in R' and one on those in R. F- or F'-scores of each alignment are calculated by comparison to R and R' respectively. Figure 4 shows the performance of the 36 domain filters on R and R'. We observe that our assumption seems valid: there is a high correlation between performance on R and R'. To quantify this observation, we repeat the process 15 times. Table 1 shows the mean and standard deviation of the correlations between performance on R' and R, for the Last.fm and UMLS datasets.

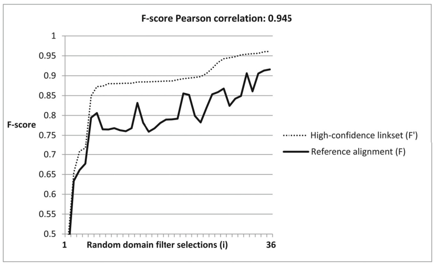


Fig. 4. Correlation between R' and R for an example run (Artists). Values have been sorted by F'_i in ascending order.

Table 1. Mean and standard deviation of the Pearson correlation $\bar{r}$ between performance on R' and R

	Last.fm	UMLS
$\bar{r}$	0.978	0.980
sd	0.009	0.007

Metaheuristic: Genetic Algorithm. We adapt our problem to a format that can be processed by a genetic algorithm. First, low-occurrence pruning (LDP) is applied to limit the search space to a processable size. The resulting domain D is then encoded as a chromosome, i.e. a bit array, where each bit represents whether the corresponding class/category is included or not. As fitness function to check the quality of a chromosome (domain filter) D, we generate a domain-restricted alignment for the source entities contained in high-confidence linkset R' and calculate F' with regard to R'.

Since discovering the best starting values is known to be very hard [3], we decide to stick with a starting population of 100 chromosomes, and the default values for selection, crossover and mutation that came with the out-of-the-box genetic algorithm implementation employed for this experiment[2]. The algorithm is run for a relatively high number of 50 generations, so as to provide the algorithm ample time to converge to an optimum (we are not concerned with running times considering our type of matching is a one-time process). The fittest remaining solution is chosen as the result.

Problem-Specific Heuristic: Broadness-Based Domain Optimization (BDO). After applying both LDP and HDP on the initial domain selection as described in section 3.3, we can try to optimize what remains of the domain selection D by selectively removing those classes that have a high probability of being too broad. One by one, a class or category is removed from D, and each time the performance F' is calculated. If performance improves, this class/category is left out. Else, it is included in D again. To maximize the effectiveness of this strategy, the order in which we remove classes/categories should be from broad to narrow. Hence D is first sorted by broadness in descending order. The size of a class, i.e. the number of instances that belong to this class (including its descendants), is taken as a measure of its broadness.

We repeat this algorithm for different settings for r_d, r_y, t_c and t_f of the HDP step, keeping track of the domain selection corresponding to the highest F'-score seen so far. The resulting D giving the highest F'-score according to R' is chosen.

Problem-Specific Heuristic: Precision-Based Domain Optimization (PDO). For this approach, we do not apply HDP, but take the resulting D from the LDP step as input. The next step is to test for each individual class/category in D how many entities are contained. This is done by generating alignment instances for the source entities of R' restricted to single-class domain filters, i.e. $|D_{single}| = 1$. Each class/category in D is tested, and for each the *precision* score is stored. Generally, the higher the precision score, the less false positive matches we find and the more relevant the class/category is to our domain of interest. In other words, the higher a precision score, the higher the probability that the class under test is specific to our domain of interest (e.g. restricting to a class "British2000sMusicalGroups" may give a high precision score when the domain of interest is "musical artists"), and vice versa. Therefore we can optimize the domain selection by applying an F'-based hillclimbing on the list of classes sorted by precision score in descending order. First, we put the class with the highest precision in domain filter D_{test} and calculate F'-score according to R'. Then we add the second class down the list to D_{test}. We check F' again, keeping track of F'_{max}, i.e. the maximum F'-score seen, and its associated domain filter D_{max}. F'_{max} and D_{max} are updated every time that $F'_{current} \geq F'_{max}$: even if F'_{max} remains the same, we prefer the largest domain filter. Once we have gone through the entire list, ending with $D_{test} = D$, the current D_{max} is our final domain filter that we take to the matching phase.

[2] http://jgap.sourceforge.net/

3.5 Domain-Aware Matching Phase

Once we have derived a suitable domain filter, the actual matching is performed. We make use of the functionality provided by DBpedia Spotlight. With the assumption that there is now a filter that tightly encapsulates the domain of the source entities to a domain in DBpedia, the context similarity and ambiguity requirements can be loosened, since disambiguation is now largely performed by this filter. We attempt to match the full source entity label(s) to a candidate in DBpedia that fits the domain filter.

4 Evaluation

Evaluation of ontology matching approaches is typically done by comparison to a baseline approach or to other matching systems that perform the same task. Since we are interested in the added value of using domain knowledge, the evaluation will primarily be done with regard to a baseline approach. We compare all domain optimization methods to each other and to two baselines that do not use domain information. Performance is evaluated by comparison to manually created reference alignments to obtain precision, recall and F-scores. All experiments are performed three-fold – once on the Last.fm dataset, and once for the Pathologic and Physiologic Function classes in UMLS.

4.1 Evaluation Strategy

In this section we list all approaches we test. We divide the approaches into three categories: baselines, basic strategies and the heuristic approaches described in section 3.4.

Baselines. As a baseline, we use DBpedia Spotlight without considering a domain filter at all. Such a baseline is essential in showing the added value of a domain-aware matching approach. However, depending on the dataset, the optimal settings of Spotlight might vary. For the purpose of this evaluation, aside from a *true baseline*, we will also artificially derive an *optimized baseline*. For the true baseline, Spotlight's settings are kept consistent with those of the the domain-aware approaches, meaning we do not rely on similarity thresholds. For the optimized baseline, Spotlight's matching parameters are optimized based on F-score according to R. This provides a theoretical upper-bound on performance for matching with Spotlight without a domain filter.

Basic Strategies. Two light-weight, basic strategies are discussed that may obtain a good result for relatively little computation time. We introduce an *optimized random selection* strategy, and an *optimized pruning* strategy – the latter applies the pruning strategies of section 3.3 in a way that we can optimize the result.

Optimized random selection. A very basic optimization approach is to take the best result from a random sample. The low-occurrence domain pruning (LDP) strategy is executed to obtain an initial domain selection. From this domain, we generate 100 sub-selections randomly by including/excluding each class/category with a probability of $\frac{1}{2}$. We calculate the F'-score of each and choose the best performer as the domain filter.

Optimized pruning. We apply both the LDP and high-generality domain pruning (HDP) strategies with different parameter settings, and choose as domain filter the result that provides the highest F'-score. This relatively cheap approach effectively deals with the high number of parameters of the pruning strategies.

Heuristics-Based Approaches. In section 3.4, we introduced one metaheuristic, the *genetic algorithm*, and two problem-specific heuristics, broadness-based domain optimization (*BDO*) and precision-based domain optimization (*PDO*), that we can apply to the derivation of a domain filter. The genetic algorithm and PDO are applied just as described in that section. The BDO approach is applied as an additional step to the optimized pruning basic strategy described before.

4.2 Experimental Results

The main results for all methods and scenarios are summarized and listed in this section. F' results are included because they provide insight into how each optimization method works. As explained before, while the correlation between F' and F has been shown to be very high, it is not a *perfect* correlation, so an increase in F' does not necessarily translate to an increase in F. Aside from the matching scores, we also list computation complexity in terms of the (estimated) number of alignments for the (1000) entities in R' we need to generate on average for each approach.

Table 2. Results for each approach on the Last.fm artist dataset, sorted by F-score

Last.fm Artist

Approach	F'	*Precision*	*Recall*	F	Complexity
PDO	0.884	0.964	0.889	0.925	~ 70
Genetic algorithm	0.889	0.960	0.868	0.912	8600
Optimized pruning	0.864	0.935	0.879	0.906	108
BDO	0.890	0.965	0.837	0.897	~ 1080
Optimized baseline	n/a	0.798	0.798	0.798	n/a
Random selection	0.768	0.722	0.865	0.787	100
True baseline	n/a	0.673	0.857	0.754	n/a

The resulting virtual F-score F' and actual precision *Precision*, recall *Recall* and F-scores F for each approach on the Last.fm dataset are displayed in table 2. Note that for complexity, there may be some variability depending on the chosen parameters, in which case an estimation is given, denoted by "$\sim$". The value for the genetic algorithm is based on the selection, crossover and mutation rate per generation, for 50 generations.

For Last.fm, a clear improvement can be seen over the baseline for any of the domain filter-based approaches barring a random selection. The best performing approach is the precision-based optimization, giving an F-score of 0.925, which is $0.925 - 0.798 = 0.127$ (12.7%) higher than the optimized baseline and $0.925 - 0.754 = 0.171$ (17.1%) higher than the true baseline in terms of absolute F-score. For reference, the domain filter that yielded this best score consists of the following classes/categories: *DBpedia classes:* `Band`. *YAGO classes:* `MusicalOrganization`, `2000sMusicGroups`. *Categories:* `Musicians`, `Musicians_by_nationality`, `Musicians_by_genre`. *Freebase classes:* `music/musical_group`, `music/group_member`, `music/artist`.

Differences amongst the optimization approaches are relatively small – the most significant difference exists between PDO and BDO ($0.925 - 0.897 = 0.025$ (2.8%)). Additionally, it shows that a higher F' does not always lead to a higher F when differences between F' are small, which suggests that optimization on an automatically created set of links works, but not perfectly.

Table 3. Summarized results for the UMLS datasets

UMLS Pathologic Function

Approach	F'	*Precision*	*Recall*	F	Complexity
Genetic algorithm	0.686	0.974	0.931	0.952	8600
PDO	0.674	0.964	0.936	0.950	~ 70
Optimized baseline	n/a	0.960	0.941	0.950	n/a
True baseline	n/a	0.950	0.941	0.946	n/a
Random selection	0.670	0.955	0.931	0.943	100
Optimized pruning	0.672	0.994	0.877	0.932	108
BDO	0.679	1.000	0.852	0.920	~ 1080

UMLS Physiologic Function

Approach	F'	*Precision*	*Recall*	F	Complexity
PDO	0.741	0.965	0.925	0.945	~ 70
Genetic algorithm	0.758	0.979	0.891	0.933	8600
BDO	0.758	0.979	0.891	0.933	~ 1080
Optimized pruning	0.753	0.981	0.883	0.930	108
Random selection	0.719	0.902	0.928	0.915	100
Optimized baseline	n/a	0.852	0.975	0.909	n/a
True baseline	n/a	0.837	0.975	0.901	n/a

The summarized results for the UMLS datasets "Pathologic Function" and "Physiologic Function" are displayed in table 3. For Pathologic Function, there is no clear difference between any of the approaches. This can be attributed to the fact that the entities in this class are highly specific terms (e.g. medical terminology for diseases) that are very unambiguous, so that correct matches to DBpedia resources are easily found. Here, we see that a domain filter can potentially even harm performance compared to the baseline. This is because entities that do not have very specific class information in DBpedia still yield a correct match in the baseline approaches due to the unambiguity of the concept name, but may get excluded from a class based domain filter.

For Physiologic Function, there is a clear improvement over the baselines again, although it is less significant than in the Artist case. This dataset contains more ambiguity than the Pathologic Function dataset (e.g. common mental processes such as "Recognition," and genes with abbreviations that are commonly used to refer to other things), but is still quite specialized, therefore the impact of a domain restriction is not as obvious as for very ambiguous entity labels such as artist and band names.

The overall results show that relying solely on the F'-scores, choosing whichever method provided the highest value, may not always be the best option – the type of approach used seems to play a role as well. The precision-based optimization approach performs quite well, being the best performer on the Artist and Physiologic Function datasets and the second best performer on the Pathologic Function dataset. This despite the fact that it never attains the highest F' compared to the other approaches. It is also the fastest approach, requiring roughly 70 alignment generations.

Statistical Significance of the Results. The precision and recall scores above are based on only a sample of all matches that our approach could produce. To extrapolate sample evaluations to statements about general performance, [6] shows that if p is the true proportion of matches that is correct (which is unknown), n the size of the sample used to approximate p (i.e. the size of the reference alignment) and $\hat{P}$ the approximation of p based on the reference alignment, then this approximation lies in the interval

$$\hat{P} \in [p-\delta, p+\delta] \text{ where } \delta = \frac{1}{\sqrt{n}} \tag{2}$$

with 95% confidence. δ is thus the 95% confidence margin of error for the result. If one result falls within this range of another result, then they do not differ sufficiently from each other to state with certainty that one is better than the other. We apply this calculation on the F-scores for each approach to verify their statistical significance.

For the Last.fm dataset, we have $n = 1000$, so the margin of error δ becomes $\frac{1}{\sqrt{1000}} = 0.032$. We can therefore conclude that all optimization approaches aside from the random selection yield a significant improvement over the baselines – the smallest difference is between the optimized baseline and the broadness-based approach, which is $0.897 - 0.798 = 0.099$. Differences among optimization approaches are at most 0.028 (between precision-based and broadness-based), so we cannot state that one optimization approach is better than the other with 95% confidence.

For the UMLS classes, we are dealing with reference alignments of size 500 for both, so here the 95% confidence margin of error for both is $\frac{1}{\sqrt{500}} = 0.045$. For Pathologic Function, none of the optimization approaches are significantly better than the baselines. For Physiologic Function, the largest difference is between the true baseline and precision-based optimization, which is a difference of 0.044. This would be borderline insignificant if we considered this as an individual occurrence, but we can see that *each* optimization approach in fact outperforms the baseline by at least 0.029. Since the optimization approaches consistently perform better, this suggests that the improvements over the baseline are significant.

5 Related Work

There has been a great amount of research done into the topic of ontology matching from the perspective of a variety of fields, such as linguistics, AI and knowledge management. As a result, a large number of ontology matching tools and methods have emerged in recent years. Surveys on the current state-of-the-art and future challenges of ontology matching have been described in [4][16]. A great majority of these systems focus on the use of string similarity mechanisms in order to determine correspondences between entities of two different ontologies or datasets – the labels of both entities might be compared in a strict or fuzzy way [22], or additional properties such as aliases or synonyms might be inferred from a background ontology such as WordNet [15][5], or translations of labels could be considered [21]. Other than property comparison, one might turn to look at the schema in which entities in both ontologies are ordered, such as semantic classes and super-classes, or relations with other entities within the same ontology [19]. In other words, this type of approach to ontology matching exploits the hierarchy of the ontology in order to determine links between entities. However, it is often tricky to derive these types of correspondences, as there is usually a string similarity matching step involved – in this case between the names of classes.

In this paper, we derived the domain of entities by bootstrapping the matching process – we generated a sample of high-confidence matches to DBpedia, then derived and optimized a domain filter from this sample. We focus on discussing related work that involves either (1) the matching of entities to DBpedia, or the use of DBpedia or other Linked Data sets as auxiliary knowledge base, (2) the incorporation of domain knowledge into the linking process, or (3) the (parameter) optimization of matchers.

The work in [12] matches entities from a database of the BBC to DBpedia by finding candidates in DBpedia based on string-similarity and wiki inlink metrics, then comparing classifications of their own entities to the classes and categories of the DBpedia candidate resources. In our case, we do not make use of classification of our source data, as this is information not always available (this is the case for Last.fm artists).

Several methods have been proposed that leverage knowledge from DBpedia or other Semantic Web sources to derive links between arbitrary Linked Data sets. BLOOMS+ [9][10] is an ontology matching system that uses the Wikipedia category hierarchy to bootstrap the process of finding schema-level links between Linked Data sets. We exploit the Wikipedia category hierarchy in a similar fashion; not to find matches directly but to find categories (and classes) that effectively describe our domain of interest. In [18], a paradigm is proposed for harvesting the Semantic Web to find related background knowledge to assist in matching two ontologies. Rather than relying on a single, high-quality knowledge source, this approach aims to automatically combine multiple heterogeneous sources. Drawbacks of the approach are its use of string comparison to find related ontologies and its disregard of the quality of the ontology selected. While results are promising given the generality of the approach, it does not match up to more focused systems such as ours. However, it can be used to complement existing techniques.

In [24], English Wikipedia pages are matched to their Chinese-language counterparts in the Baidu Baike knowledge base. The authors predict correspondences by three language-independent features: in- and outlink homophily, category homophily and author interest. In essence, a "domain" is determined for each individual page, and the page that best matches this from the target dataset is selected for linking. This method requires a large set of existing links in order to train a model, however, and is therefore quite different from our scenario. Kalyanpur et. al. [11] describe a type coercion framework for use in a question answering scenario; in this case for IBM's Watson

[7]. Their system makes extensive use of DBpedia and domain knowledge extracted from DBpedia. For our approach, we obtain candidate DBpedia resources in much the same way, and similarly leverage the class/category hierarchies to improve the quality of matching. The main difference is that the Watson system attempts to find the correct candidate class by string comparison and hierarchy-based alignment with the source (question) type, while we do not assume to have source type information available.

The authors of [8] present an approach to automatically generate linkage rules from a set of reference links. A genetic algorithm is applied to learn which rules and parameters work best on a dataset, based on performance according to a partial reference alignment. They show that the automatically learned rules achieve similar accuracy to those manually created by humans. In [17], ECOMatch is proposed. ECOMatch is a generic parameterization system designed for use on top of any ontology matching system. It automatically determines a suitable parameter configuration based on user-provided example mappings. The authors show that a correlation exists between scores of a configuration according to a portion of a reference alignment and a full reference alignment. Some aspects of our approach are similar to these two works. Like the first work, we also use a genetic algorithm for optimization, but rather than learning linkage rules to find correspondences, we learn a domain filter to restrict correspondences to. ECOMatch is also different to our work in that we do not try to find an optimal configuration for a matcher. Most importantly, unlike both described works, we do not require a human-constructed set of reference links to find the most suitable domain filter.

6 Conclusion

This paper presented work on the development and evaluation of a domain-aware ontology matching approach. We showed that we can improve the quality of generated links over traditional approaches by bootstrapping the matching process – we derive a filter in a fully unsupervised way that describes and delimits the domain of the source dataset in terms of classes and categories collected from DBpedia.

We showed that when the dataset we want to match to DBpedia is particularly ambiguous, such as is the case for Last.fm artists, a significant gain in matching quality can be obtained by first bootstrapping these matches and deriving a domain filter to restrict them. With the best-performing approach, a 17.1% improvement in F-score was gained over a baseline, domain-unaware approach. For the UMLS "Physiologic Function" subdomain, which is a much more specialized dataset but still contains some ambiguous terms, we still saw an improvement of 4.4%. On the other hand, given an extremely specialized and unambiguous dataset such as the "Pathologic Function" subdomain of the UMLS, matching with a domain filter does not provide a significant improvement, and can potentially even hurt quality.

Future Work. Our method is currently fixed to DBpedia as target ontology, and DBpedia Spotlight as matcher, but could theoretically work in a generic ontology matching system as well. Future work would be to generalize the current system to also support different target ontologies, or to integrate our approach as a separate step into already existing ontology matching systems. This would also allow for evaluation of our method with regard to existing systems.

Secondly, due to this reliance on DBpedia Spotlight – a tool catered to the annotation of free text – certain valid surface forms are not recognized due to being too common. We expect that creating a custom parser for Spotlight that is specialized towards ontology matching could significantly improve the absolute matching scores.

Acknowledgements. This work is partly funded by the FP7 projects EURECA under grant nr. 288048 and ImREAL under grant nr. 257831.

References

1. Benjamins, V.R., Contreras, J., Corcho, O., Gómez-pérez, A.: Six Challenges for the Semantic Web. In: KR 2002 Semantic Web Workshop, vol. 1 (2002)
2. Bizer, C., Heath, T., Berners-Lee, T.: Linked Data - The Story So Far. International Journal on Semantic Web and Information Systems 5(3), 1–22 (2009)
3. Eiben, A.E., Hinterding, R., Michalewicz, Z.: Parameter Control in Evolutionary Algorithms. IEEE Transactions on Evolutionary Computation 3(2), 124–141 (1999)
4. Euzenat, J., Shvaiko, P.: Ontology matching, 1st edn. Springer (July 2007)
5. Giunchiglia, F., Yatskevich, M., Shvaiko, P.: Semantic matching: Algorithms and implementation. Technical report, University of Trento (January 2007)
6. Hage, W.V., Isaac, A., Aleksovski, Z.: Sample evaluation of ontology-matching systems. In: Fifth Int. Workshop on Evaluation of Ontologies and Ontology-based Tools, ISWC 2007 (2007)
7. IBM. Watson (retrieved on December 14, 2011)
8. Isele, R., Bizer, C.: Learning linkage rules using genetic programming. In: Proceedings of the Sixth International Workshop on Ontology Matching at ISWC 2011, p. 13 (2011)
9. Jain, P., Hitzler, P., Sheth, A.P., Verma, K., Yeh, P.Z.: Ontology Alignment for Linked Open Data. 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. 402–417. Springer, Heidelberg (2010)
10. Jain, P., Yeh, P.Z., Verma, K., Vasquez, R.G., Damova, M., Hitzler, P., Sheth, A.P.: Contextual Ontology Alignment of LOD with an Upper Ontology: A Case Study with Proton. 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. 80–92. Springer, Heidelberg (2011)
11. 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)
12. 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)
13. Luke, S.: Essentials of Metaheuristics. Lulu.com (March 2011)
14. Mendes, P.N., Jakob, M., García-Silva, A., Bizer, C.: Dbpedia spotlight: Shedding light on the web of documents. In: Proc. of the 7th Intl. Conference on Semantic Systems (2011)
15. Miller, G.A.: WordNet: a lexical database for English. Commun. ACM 38(11), 39–41 (1995)

16. Pavel, S., Euzenat, J.: Ontology matching: State of the art and future challenges. IEEE Transactions on Knowledge and Data Engineering 99(PrePrints), 1 (2012)
17. Ritze, D., Paulheim, H.: Towards an automatic parameterization of ontology matching tools based on example mappings. In: Proceedings of the Sixth International Workshop on Ontology Matching at ISWC 2011, vol. 814, p. 37. CEUR-WS (2011)
18. Sabou, M., d'Aquin, M., Motta, E.: Exploring the Semantic Web as Background Knowledge for Ontology Matching. In: Spaccapietra, S., Pan, J.Z., Thiran, P., Halpin, T., Staab, S., Svatek, V., Shvaiko, P., Roddick, J. (eds.) Journal on Data Semantics XI. LNCS, vol. 5383, pp. 156–190. Springer, Heidelberg (2008)
19. Shvaiko, P., Euzenat, J.: A Survey of Schema-Based Matching Approaches. In: Spaccapietra, S. (ed.) Journal on Data Semantics IV. LNCS, vol. 3730, pp. 146–171. Springer, Heidelberg (2005)
20. Slabbekoorn, K., Hollink, L., Houben, G.-J.: Domain-aware matching of events to dbpedia. In: Proc. of the DeRiVE 2011 Workshop, ISWC 2011, vol. 779. CEUR-WS (2011)
21. Spohr, D., Hollink, L., Cimiano, P.: A Machine Learning Approach to Multilingual and Cross-Lingual Ontology Matching. 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. 665–680. Springer, Heidelberg (2011)
22. Stoilos, G., Stamou, G., Kollias, S.: 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)
23. Suchanek, F.M., Kasneci, G., Weikum, G.: Yago: a core of semantic knowledge. In: Proceedings of WWW 2007, pp. 697–706 (2007)
24. Wang, Z., Li, J., Wang, Z., Tang, J.: Cross-lingual knowledge linking across wiki knowledge bases. In: Proceedings of WWW 2012, pp. 459–468. ACM (2012)

Rapidly Integrating Services into the Linked Data Cloud

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. The amount of data available in the Linked Data cloud continues to grow. Yet, few services consume and produce linked data. There is recent work that allows a user to define a linked service from an online service, which includes the specifications for consuming and producing linked data, but building such models is time consuming and requires specialized knowledge of RDF and SPARQL. This paper presents a new approach that allows domain experts to rapidly create semantic models of services by demonstration in an interactive web-based interface. First, the user provides examples of the service request URLs. Then, the system automatically proposes a service model the user can refine interactively. Finally, the system saves a service specification using a new expressive vocabulary that includes lowering and lifting rules. This approach empowers end users to rapidly model existing services and immediately use them to consume and produce linked data.

Keywords: linked data, linked API, service modeling.

1 Introduction

Today's Linked Open Data (LOD) cloud consists primarily of databases that have been translated into RDF and linked to other datasets (e.g., DBpedia, Freebase, Linked GeoData, PubMed). Often, information is not current (e.g., Steve Jobs was listed as CEO of Apple Computer in DBpedia for months after his passing), and timely information is not available at all (e.g., the LOD cloud has no information about the current weather or events for any city).

Web APIs provide the opportunity to remedy this problem as there are thousands of Web APIs that provide access to a wealth of up-to-date data. For example, the programmableweb[1] lists over 6,000 APIs that provide data on an immense variety of topics. The problem is that most of these APIs provide data in JSON and XML and are not in any way connected to the LOD cloud. For example, in the programmableweb only 65 APIs (about 1%) provide information in RDF, and the rest provide information in XML and JSON. Our goal is to make it easy to connect the remaining thousands of XML and JSON-based

[1] http://www.programmableweb.com/apis

Web APIs to the LOD cloud. To do so we need to make it easy to represent the semantics of Web APIs in terms of well known vocabularies, and we need to wrap these APIs so that they can consume RDF from the LOD cloud and produce RDF that links back to the LOD cloud.

Several approaches have been developed to integrate Web APIs with the Linked Data cloud. One approach is to annotate the service attributes using concepts of known ontologies and publish the service descriptions into the cloud [10,11]. This allows users to easily discover relevant services. A second approach is to wrap the APIs to enable them to communicate at the semantic level so that they can consume and produce linked data [9,5]. A third approach is to create a uniquely identifiable resource for each instance of an API invocation and then link that resource to other data sets [13,12]. Using this approach it is possible to invoke the API by dereferencing the corresponding resource URI.

The main obstacle preventing these approaches from gaining wide acceptance is that building the required models is difficult and time-consuming. In these approaches, a developer needs to create a model that defines the mapping from the information consumed and produced by Web APIs to Semantic Web vocabularies. In addition, the developer must also write the lowering and lifting specifications that lower data from RDF into the format expected by the Web APIs and then lift the results of the Web API invocations to RDF. Writing these models and the required lowering and lifting specifications often requires in-depth knowledge of RDF, SPARQL, and languages such as XPath or XSLT[2].

The key contribution of our work is a method to semi-automatically build semantic models of Web APIs, including the lowering and lifting specifications. In our approach, users provide sample URLs to invoke a service and the vocabularies they want to use to model the service. The system automatically invokes the service and builds a model to capture the semantics of the inputs, outputs and relationships between inputs and outputs. The system also provides an easy-to-use web-based graphical interface through which users can visualize and adjust the automatically constructed model. The resulting models, represented in RDF using standard vocabularies, enable service discovery using SPARQL queries. In addition, our system can automatically generate the lowering and lifting specifications from these models so that the Web APIs become immediately executable and able to consume and produce linked data.

2 Overview

The objective of our approach is to create linked APIs by combining traditional Web APIs and the Linked Data cloud in two aspects. We want to publish semantic service descriptions into the cloud that can be used by the linked data community in service discovery and composition. We also want to deploy APIs that interact at the semantic level, directly consuming Linked Data and generating RDF data that is linked to the input data. Since most Web APIs use the

[2] Extensible Stylesheet Language Transformations.

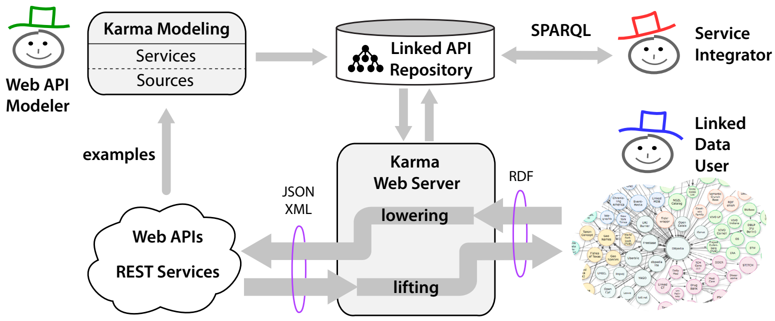


Fig. 1. The overview of our approach to create linked APIs

HTTP GET method, in this paper, we focus on these APIs and assume that all inputs for service invocation are embedded in the invocation URL. The approach presented in this paper builds upon and extends Karma [3,14,15], our modeling and integration framework.[3]

Figure 1 shows the three main steps of our approach. The first step [14], the foundation of the rest of the process, is to semi-automatically build a service model that represents the semantics of the API functionality (Section 3). Users first provide examples of service request URLs. Karma then invokes the services and constructs a worksheet that contains both the inputs and the outputs produced. In this step we leverage our prior work on modeling sources, using it to construct a model of the input/output worksheet. The process is fast because users only need to provide a few examples of service request URLs and then adjust the automatically generated model. The process is also easy because users interact with the system through a graphical user interface and are not required to know Semantic Web technologies such as RDF, SPARQL, and XSLT.

The second step is to formally represent the semantic models built in the previous step (Section 4). Once the user models the API, Karma automatically generates the service descriptions and stores them in a repository. The linked API repository provides a SPARQL interface that service integrators can use for service discovery. For example, a service integrator can issue a query to retrieve all the APIs that return neighborhood information given latitude and longitude. A service integrator can also employ reasoning algorithms to generate a plan to achieve a specific goal.

The final part of our method is to deploy linked APIs on a Web server where they can be directly invoked, consuming and producing RDF (Section 5). The Web server provides a REST interface that Linked Data users can use to retrieve RDF data. It uses the service descriptions in the repository to automatically lower the input RDF, to invoke the actual Web API, and to lift the output to return linked (RDF) data.

[3] This paper is a significantly extended version of a workshop paper [14].

3 Semi-automatically Modeling Web APIs

Our approach to model Web APIs using Karma consists of two parts. In the first part, users provide Karma a collection of sample invocation URLs. Karma uses these URLs to invoke the APIs and construct a worksheet that contains the inputs and corresponding outputs for each service invocation. In the second part, we use our prior Karma work to construct a model of the resulting worksheet. In this section we describe our prior Karma work, describe the procedure to construct the inputs/outputs worksheet from the invocation URLs, and illustrate the whole process using an example.

3.1 Previous Work on Source Modeling

To provide a better understanding of our new work on service modeling, in this section we briefly review our previous work on source modeling [3]. The process of modeling sources in Karma is a semi-automatic process that is initiated when users load a data source. Karma supports importing data from various structured sources including relational databases, spreadsheets, JSON, and XML. Then, users specify the vocabularies to which they want to map the source, and Karma automatically constructs a model that users can adjust. The output is a formal model that specifies the mapping between the source and the target ontology. Specifically, Karma generates a GLAV source description [3,7].

The modeling process consists of two steps. The first step is to characterize the type of data by assigning a *semantic type* to each column. In our approach, a semantic type can be either an OWL class or the range of a data property (which we represent by the pair consisting of a data property and its domain). We use a conditional random field (CRF) [6] model to learn the assignment of semantic types to columns of data [2]. Karma uses this model to automatically suggest semantic types for data columns. If the correct semantic type is not among the suggested types, users can browse the ontology through a user friendly interface to find the appropriate type. Karma automatically re-trains the CRF model after these manual assignments.

The second part of the modeling process is to identify the relationships between the inferred semantic types in the ontology. Given the domain ontology and the assigned semantic types, Karma creates a graph that defines the space of all possible mappings between the source and the ontology [3]. The nodes in this graph represent classes in the ontology, and the links represent properties. The mapping is not one to one, because there might be several instances of the same class present in the source.

Once Karma constructs the graph, it computes the source model as the minimal tree that connects all the semantic types. The minimal tree corresponds to the most concise model that relates all the columns in a source, and this is a good starting point for refining the model. We use a Steiner tree algorithm to compute the minimal tree. Given an edge-weighted graph and a subset of the vertices, called Steiner nodes, the goal is to find the minimum-weight tree in the graph that spans all the Steiner nodes. The Steiner tree problem is NP-complete, but we use a heuristic algorithm [4] with an approximation ratio bounded by $2(1 - 1/l)$, where l is the number of leaves in the optimal Steiner tree.

It is possible that multiple minimal trees exist, or that the correct interpretation of the data is captured by a non-minimal tree. In these cases, Karma allows the user to interactively impose constraints on the algorithm to build the correct model. Karma provides an easy-to-use GUI in which the user can adjust the relationships between the source columns [3].

3.2 Service Invocation

To enable Karma to model services in the same way that it models structured sources, we need to generate a table of example inputs and outputs. To this end, Karma asks the user to provide samples of the Web API requests. Karma parses the URLs and extracts the individual input parameters along with their values. For each request example, Karma invokes the service and extracts the output attributes and their values from the XML or JSON response. At the end, Karma joins the inputs and the outputs and shows them in a table.

Once this table is constructed, we apply our prior Karma work on source modeling to construct a model of the table. As described above, our source modeling technique captures the relationships among all columns of a source. Consequently, when we apply it to the table constructed from the API invocation URLs, the resulting model will capture the relationships between the inputs and outputs of the API.

An alternative method to collect examples of the API inputs and outputs is to extract such information from the documentation pages of the APIs. According to a comprehensive study on Web APIs by Maleshkova et al. [8], 83.8% of the APIs indexed in `programmableweb` provide a sample request and 75.2% of them also provide a sample response. In future work we plan to mine these documentation pages to extract examples of inputs and outputs.

3.3 Example

We illustrate our service modeling approach with an example from the GeoNames APIs[4]. We model the `neighbourhood` API,[5] which takes the latitude and longitude of a geographic feature as input and returns information about the neighborhood of that feature. To keep the example concise, we only consider the region name, the nearby city, the country code, and the country name. An example of the API invocation URL and the API response are shown in Figure 2(a) and (b). Figure 2(c) shows the table of inputs and outputs that Karma constructs using the invocation URLs listed in the first column of the table.

In the next step, Karma treats the service table as a data source and maps it to the ontologies given by the user (in our example GeoNames[6] and WGS84[7]

[4] `http://www.geonames.org/export/ws-overview.html`

[5] `http://www.geonames.org/export/web-services.html#neighbourhood`

[6] `http://www.geonames.org/ontology/ontology_v3.01.rdf`

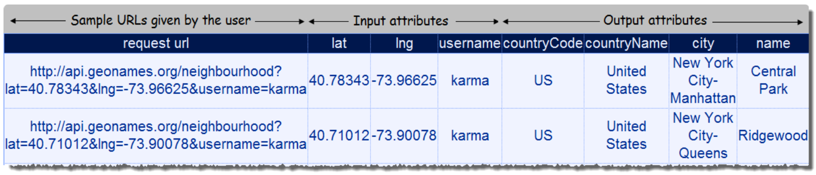
[7] `www.w3.org/2003/01/geo/wgs84_pos`

```
http://api.geonames.org/neighbourhood?lat=40.78343&lng=-73.96625&username=karma
```

(a) Service Invocation URL.

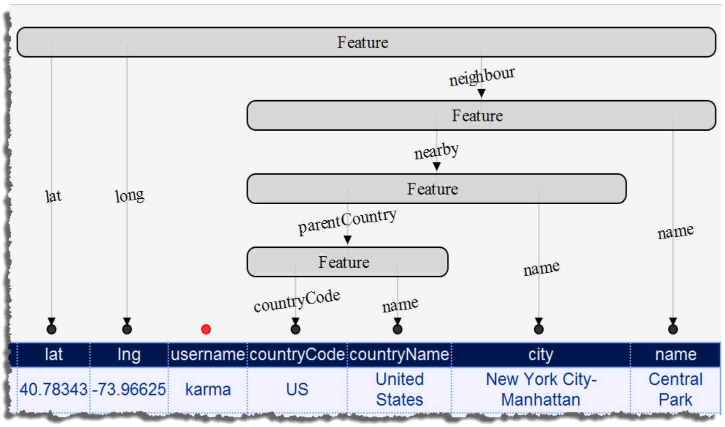
```
<geonames><neighbourhood>
      <countryCode>US</countryCode>
      <countryName>United States</countryName>
      <city>New York City-Manhattan</city>
      <name>Central Park</name>
       ...
</neighbourhood></geonames>
```

(b) Service response (XML).



Sample URLs given by the user	Input attributes			Output attributes			
request url	lat	lng	username	countryCode	countryName	city	name
http://api.geonames.org/neighbourhood?lat=40.78343&lng=-73.96625&username=karma	40.78343	-73.96625	karma	US	United States	New York City-Manhattan	Central Park
http://api.geonames.org/neighbourhood?lat=40.71012&lng=-73.90078&username=karma	40.71012	-73.90078	karma	US	United States	New York City-Queens	Ridgewood

(c) The user provides examples of the service requests. Karma extracts the input parameters, invokes the API, extracts the output attributes from the invocation response, and joins the outputs with the input data in one table.



lat	lng	username	countryCode	countryName	city	name
40.78343	-73.96625	karma	US	United States	New York City-Manhattan	Central Park

(d) Screenshot showing the service model in Karma.

neighbourhood($lat, $long, @countryCode, @countryName, @city, @name) →
gn:Feature(v1) ∧ wgs84:lat(v1, $lat) ∧ wgs84:long(v1, $long) ∧ gn:*neighbourhood*(v1, v2) ∧
gn:Feature(v2) ∧ gn:name(v2, @name) ∧ gn:*nearby*(v2, v3) ∧
gn:Feature(v3) ∧ gn:name(v3, @city) ∧ gn:*parentCountry*(v3, v4) ∧
gn:Feature(v4) ∧ gn:countryCode(v4, @countryCode) ∧ gn:name(v4, @countryName)

(e) Logical LAV rule representing the semantics of the neighbourhood API. Input and output attributes are marked with $ and @ respectively. The API is described with terms from two ontologies: GeoNames (gn:) and WGS84.

Fig. 2. Karma service modeling process: (a) Web API invocation URL, (b) XML response, (c) Web API inputs and outputs in Karma's interface, (d) Semantic model in Karma's interface, and (e) formal model as a LAV rule

ontologies). Karma automatically recommends the top four most likely semantic types (a class or a data-property/domain pair) for each column and the user assigns the semantic type by either selecting one of the suggested types or choosing another type from the ontology. After each assignment, Karma uses its Steiner Tree algorithm to recompute the tree that relates the semantic types.

The final model is shown in Figure 2(d). The service inputs (`lat` and `lng`) are mapped to the *lat* and *long* properties of a *Feature*. This feature is related via the *neighbor* object property to another *Feature*, which in turn is related to other geographical features that describe the remaining outputs (the nearby city and the parent country). Users can change the semantic types by clicking on the black circles above the column names. They can also adjust the other elements of the model by clicking on the property names (labels on the arrows) to select alternative properties or to choose alternative domains for those properties.

Figure 2(e) shows the LAV rule that captures the formal semantics of the service model that was shown in graphical form in Figure 2(d). In this rule, *Feature* is a class and *neighbourhood*, *nearby*, *name*, and *parentCountry* are properties in the GeoNames ontology, and *lat* and *long* are properties in WGS84 ontology.

4 Building a Linked API Repository

To integrate Web APIs to the Linked Data Cloud we publish an RDF representation of the API models in the Linked API Repository. In this section, we describe how we represent the linked APIs in the repository and how these declarative representations provide support for service discovery and composition.

4.1 Representing Linked APIs

Our models of Web APIs represent both the syntax and the semantics of the API. The syntactic part provides the information needed to invoke the service, including address URL, HTTP method, access credentials, and input parameters. The semantic part represents the types of the inputs and the outputs and the relationship among them in terms of a target vocabulary (ontology).

There are several vocabularies to represent linked services. WSMO-Lite[8] and Minimal Service Model[9] (MSM) are RDF vocabularies that can represent the syntax of Web APIs, but can only partially represent the semantics. They can represent the types of the inputs and outputs using terms in ontologies, but they cannot represent the relationships among them. Other approaches [5,12] use SPARQL graph patterns to define the inputs and outputs. They can model relationships among inputs and outputs, but discovery is difficult as there are no standard facilities to query graph patterns.

We introduce an expressive ontology that extends and combines the strengths of the existing vocabularies in one model. It can represent the semantics of

[8] `http://www.w3.org/Submission/WSMO-Lite/`
[9] `http://cms-wg.sti2.org/minimal-service-model/`

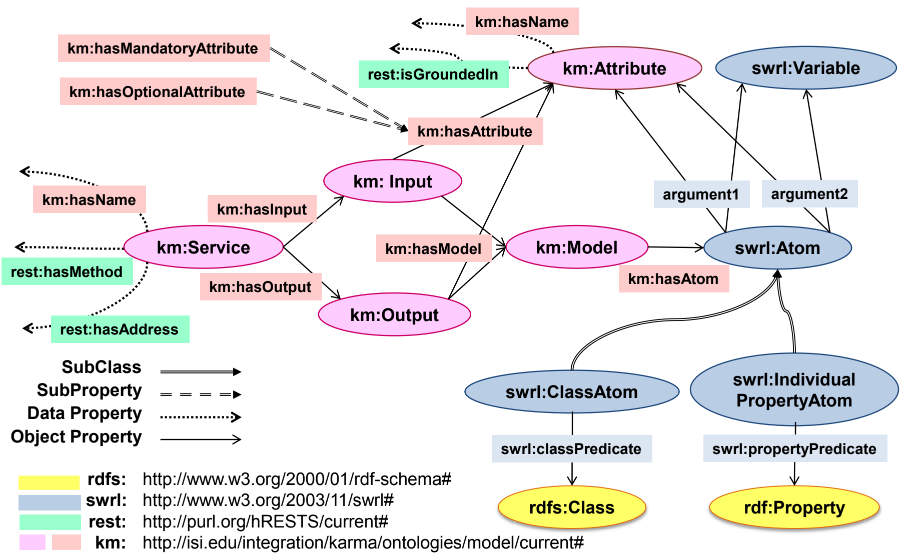


Fig. 3. The ontology that we use to formally describe Web APIs

services including relationships among inputs and outputs, and it uses RDF(S) so that models can be queried using SPARQL.

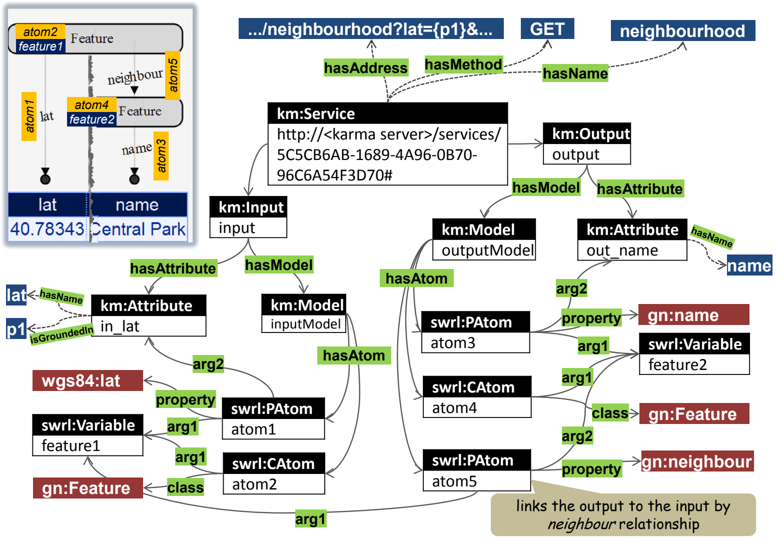
Figure 3 shows our ontology for modeling Web APIs. We re-use the SWRL[10] vocabulary to define the input and output model. In this ontology, both input and output have a *Model* which includes one or more *Atom* instances. A *ClassAtom* shows the membership of an instance to a class and an *IndividualPropertyAtom* describes an instance of a property.

We map the service semantic model (Figure 2(d)) to the introduced vocabulary by adding a *ClassAtom* for each class instance (rounded rectangles) and an *IndividualPropertyAtom* for each property (arrows). For example, to express the part of the semantic model where the top `Feature` box is connected to the the `lat` column, we first create a *ClassAtom* whose *classPredicate* has the value *gn:Feature* and its *argument1* is a new *Variable*. Then, we add an *IndividualPropertyAtom* in which the *propertyPredicate* is *wgs84:lat*, *argument1* is the same variable in the *ClassAtom*, and *argument2* is the URI of the *lat* input attribute. Figure 4 includes both graphical and N3 notation of a snippet of the `neighbourhood` API model. The input and output model is interpreted as a conjunctive formula, which is the RDF rendering of the LAV rule generated by Karma that captures the semantics of the Web API (cf. Figure 2(e))

4.2 Querying the Repository

Karma stores the API descriptions in a triple store. This linked API repository offers a SPARQL interface that can be used to discover desired APIs. Our rich

[10] Semantic Web Rule Language: `http://www.w3.org/Submission/SWRL/`



```
@prefix : <http://<karma server>/services/5C5CB6AB-1689-4A96-0B70-96C6A54F3D70#> .
@prefix gn: <http://www.geonames.org/ontology#> .
@prefix wgs84: <http://www.w3.org/2003/01/geo/wgs84_pos#> .
...
: a km:Service;
    km:hasName "neighbourhood" ;
    hrests:hasAddress "http://api.geonames.org/neighbourhood?
                       lat={p1}&lng={p2}&username={p3}" ^^
                       hrests:URITemplate ;
    hrests:hasMethod "GET"; km:hasInput :input; km:hasOutput :output.

:input a km:Input;
    km:hasAttribute :in_lat, ... ;
    km:hasModel :inputModel .
:in_lat a km:Attribute;
    km:hasName "lat" ;
    hrests:isGroundedIn
       "p1"^^rdf:PlainLiteral .
...
:feature1 a swrl:Variable .
:inputModel a km:Model;
    km:hasAtom
       [ a swrl:ClassAtom ;
       swrl:classPredicate gn:Feature;
       swrl:argument1 :feature1 ];
    km:hasAtom
       [ a swrl:IndividualPropertyAtom;
       swrl:propertyPredicate wgs84:lat;
       swrl:argument1 :feature1;
       swrl:argument2 :in_lat];
    ...

:output a km:Output;
    km:hasAttribute :out_name, ... ;
    km:hasModel :outputModel .
:out_name a km:Attribute;
    km:hasName "name" .
...

:feature2 a swrl:Variable .
:outputModel a km:Model;
    km:hasAtom
       [ a swrl:ClassAtom ;
       swrl:classPredicate gn:Feature;
       swrl:argument1 :feature2] ;
    km:hasAtom
       [ a swrl:IndividualPropertyAtom ;
       swrl:propertyPredicate gn:neighbour;
       swrl:argument1 :feature1 ;
       swrl:argument2 :feature2];
    km:hasAtom
       [ a swrl:IndividualPropertyAtom ;
       swrl:propertyPredicate gn:name ;
       swrl:argument1 :feature2 ;
       swrl:argument2 :out_name];
    ...
```

Fig. 4. A snippet of the neighbourhood API model represented both graphically and formally (N3 notation)

models support a variety of interesting queries. The following SPARQL query finds all services that take latitude and longitude as inputs. It is difficult to support this type of query in models that use SPARQL graph patterns because the patterns are represented as strings and it is difficult to reason over them.

```
SELECT ?service WHERE {
?service km:hasInput [km:hasAttribute ?i1, ?i2].
?service km:hasInput [km:hasModel [km:hasAtom
[swrl:propertyPredicate wgs84:lat;   swrl:argument2 ?i1],
[swrl:propertyPredicate wgs84:long;  swrl:argument2 ?i2]]]}
```

In the WSMO-Lite and MSM specifications, which are in RDF, inputs and outputs of a service are linked to concepts and properties in ontologies, so they support the previous example query. However, they do not support answering more complex questions that take into account the relationships between the attributes. For example, suppose a user wants to find services that return the neighborhood regions given a latitude and longitude. These models cannot be used to answer this question because they cannot represent the relationship (`neighbour`) between the inputs (`lat, lng`) and the output regions. In our model, we can write the following SPARQL query:

```
SELECT ?service WHERE {
?service km:hasInput [km:hasAttribute ?i1, ?i2].
?service km:hasOutput [km:hasAttribute ?o1].
?service km:hasInput [km:hasModel [km:hasAtom
[swrl:classPredicate    gn:Feature; swrl:argument1 ?f1],
[swrl:propertyPredicate wgs84:lat;  swrl:argument1 ?f1; swrl:argument2 ?i1],
[swrl:propertyPredicate wgs84:long; swrl:argument1 ?f1; swrl:argument2 ?i2]]].
?service km:hasOutput [km:hasModel [km:hasAtom
[swrl:classPredicate    gn:Feature;   swrl:argument1 ?f2],
[swrl:propertyPredicate gn:name;      swrl:argument1 ?f2; swrl:argument2 ?o1],
[swrl:propertyPredicate gn:neighbour; swrl:argument1 ?f1; swrl:argument2 ?f2]]]}
```

5 Deploying Linked APIs

Publishing service descriptions into the Linked Data cloud is the first step in creating linked APIs. The next step is to deploy APIs that are able to consume data directly from the cloud and also to produce linked data. One of the benefits of our service models is that they include lowering and lifting instructions that our system can directly execute. This allows the user to easily wrap existing Web APIs without writing separate lowering and lifting specifications. In this section, we describe how we set up linked APIs and how we enable them to communicate at the semantic level (RDF).

5.1 Invoking a Linked API

As shown in Figure 1, the Karma Web server is the component that enables users to invoke the linked APIs. Users communicate with this server through a REST interface to send the RDF as input and get linked data as output. If the user sends a GET request to this endpoint,[11] he will receive the service specification in RDF. Calling the linked API with a POST request is the method to feed linked data to a service. Karma performs the following steps to execute these POST requests, which include the input RDF in its body:

– The Karma Web server extracts the service identifier from the request and uses it to retrieve the model from the repository.
– The server verifies that the input RDF satisfies the input semantic model and rejects requests that are incompatible with the input model. To do this it creates a SPARQL query according to the API input model and executes it on the input data.
– The server uses the model to do lowering, creating the appropriate URL for the original Web API (section 5.2).
– The server invokes the Web API and gets the results.
– The server uses the output model to convert the output from XML/JSON to RDF and link it to the input (section 5.2).
– The server returns the linked data to the caller in RDF.

To enable callers to determine the appropriate RDF graph that can be used as the input, the Karma Web server offers an interface[12] to get the SPARQL pattern that corresponds to the service input. The caller can execute this SPARQL query on its RDF store to obtain the appropriate input graph to call the linked API.

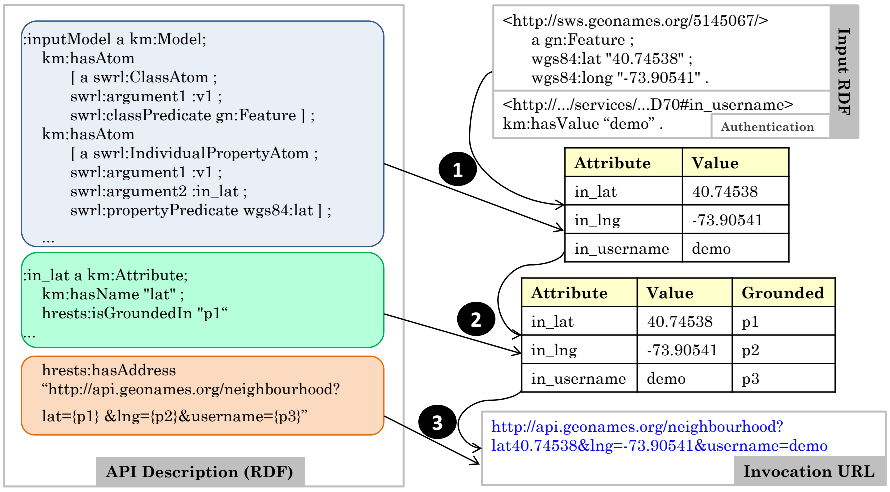
5.2 Lowering and Lifting

One of the advantages of our approach is that our models contain all the information needed to automatically execute the required lowering and lifting, obviating the need to manually write their specification. We describe the lowering and lifting processes using an example. Suppose that the server receives a POST request for the `neighborhood` API. The body of the request has RDF triples from the GeoNames data source with the coordinates of geographical features.

Figure 5 illustrates the lowering process. When the Karma Web server receives the HTTP POST request, it extracts the RDF triples from the body and retrieves the model from the repository. The *swrl:classPredicate* of the *ClassAtom* and the *swrl:propertyPredicate* of the *IndividualPropertyAtom* are URIs of the corresponding classes and properties in the input RDF (e.g., *wgs84:lat*), enabling the server to retrieve the corresponding values (e.g., `40.74538`). Every input attribute (e.g., *:in-lat*) has a *isGroundedIn* property which indicates its position in the service address. This enables the server to build the invocation URL directly from the service specification without writing an explicit lowering schema. For the input parameters such as authentication key that are not part of the semantic model, the user would provide separate statements in the input graph.

[11] The address of the REST API is `http://<karma server>/services/{id}` in which `id` is the service identifier created automatically by Karma when it generates the API description.

[12] `http://<karma server>/services/{id}/input?format=SPARQL`



```
:inputModel a km:Model;
   km:hasAtom
        [ a swrl:ClassAtom ;
        swrl:argument1 :v1 ;
        swrl:classPredicate gn:Feature ] ;
   km:hasAtom
        [ a swrl:IndividualPropertyAtom ;
        swrl:argument1 :v1 ;
        swrl:argument2 :in_lat ;
        swrl:propertyPredicate wgs84:lat ] ;
   ...
```

```
:in_lat a km:Attribute;
   km:hasName "lat" ;
   hrests:isGroundedIn "p1"
...
```

```
   hrests:hasAddress
   "http://api.geonames.org/neighbourhood?
   lat={p1} &lng={p2}&username={p3}"
```

```
<http://sws.geonames.org/5145067/>
     a gn:Feature ;
     wgs84:lat "40.74538" ;
     wgs84:long "-73.90541" .
```

```
<http://.../services/...D70#in_username>
km:hasValue "demo" .
```

Attribute	Value
in_lat	40.74538
in_lng	-73.90541
in_username	demo

Attribute	Value	Grounded
in_lat	40.74538	p1
in_lng	-73.90541	p2
in_username	demo	p3

```
http://api.geonames.org/neighbourhood?
lat40.74538&lng=-73.90541&username=demo
```

Fig. 5. Lowering the RDF data to create the invocation URL of the Web API

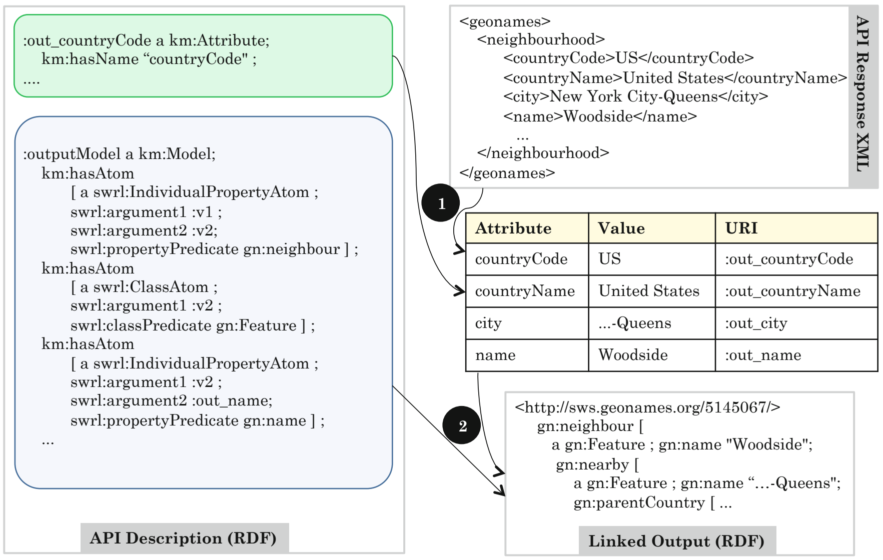
For instance, in the example in Figure 5, the user would add a triple, such as `<serviceURI:username km:hasValue "demo">`, to the input data.

In the next step, the Karma Web server links the outputs to the input RDF in order to return additional information about the inputs to the user. The service returns the outputs in XML or JSON, and these need to be converted to RDF. Figure 6 illustrates the lifting process. First, for each output attribute (e.g., *:out_countryCode*), Karma uses *km:hasName* property to get its name (e.g., `countryCode`). This name is compared to the XML tags to identify the corresponding value (e.g., `US`). Then, Karma exploits the output model of the API to create a RDF graph of the output values. If there is a variable in the output model that is also part of the input model (e.g., *:v1*), that means Karma already knows its value from the input RDF. For the other variables in the output model (e.g., *:v2*), Karma creates blank nodes according to their type (e.g., *gn:Featue*), denoted by *swrl:classPredicate* property. A video demonstrating how to model the neighbourhood API in Karma and wrap it as a linked API is available on the Karma web site[13].

6 Related Work

Three recent approaches address the integration of services and the Linked Data cloud. The first approach, called Linked Services [10,11], focuses on annotating services and publishing those annotations as Linked Data. The second approach creates services, called Linked Open Services (LOS) [9,5], that consume and produce Linked Data. Finally, Linked Data Services (LIDS) [12,13] integrates data services with Linked Data by assigning a URI to each service invocation.

[13] `http:/isi.edu/integration/karma`



```
:out_countryCode a km:Attribute;
   km:hasName "countryCode" ;
....
```

```
:outputModel a km:Model;
   km:hasAtom
        [ a swrl:IndividualPropertyAtom ;
        swrl:argument1 :v1 ;
        swrl:argument2 :v2;
        swrl:propertyPredicate gn:neighbour ] ;
   km:hasAtom
        [ a swrl:ClassAtom ;
        swrl:argument1 :v2 ;
        swrl:classPredicate gn:Feature ] ;
   km:hasAtom
        [ a swrl:IndividualPropertyAtom ;
        swrl:argument1 :v2 ;
        swrl:argument2 :out_name;
        swrl:propertyPredicate gn:name ] ;
   ...
```

```
<geonames>
  <neighbourhood>
      <countryCode>US</countryCode>
      <countryName>United States</countryName>
      <city>New York City-Queens</city>
      <name>Woodside</name>
        ...
  </neighbourhood>
</geonames>
```

Attribute	Value	URI
countryCode	US	:out_countryCode
countryName	United States	:out_countryName
city	...-Queens	:out_city
name	Woodside	:out_name

```
<http://sws.geonames.org/5145067/>
   gn:neighbour [
      a gn:Feature ; gn:name "Woodside";
      gn:nearby [
         a gn:Feature ; gn:name "...-Queens";
         gn:parentCountry [ ...
```

Fig. 6. Lifting the XML response to create linked data

The service URI is linked to resources in the Linked Data cloud and dereferencing the URI provides RDF information about the linked resources.

Linked Services uses a simple RDF ontology, called Minimal Service Model (MSM), to annotate the service and publish it as Linked Data. MSM uses the `modelReference` property of the SAWSDL vocabulary [1] to map service inputs and outputs to the concepts in ontologies. In Karma, rather than just annotating the service attributes, we also model the relationships between them in order to support more sophisticated service discovery and composition. Moreover, when it comes to consuming and producing RDF for existing Web APIs, MSM required modelers to provide explicit lowering and lifting schema using the `sawsdl:loweringSchema` and `sawsdl:liftingSchema` relations. In contrast, our modeling approach includes enough information to automatically derive the lowering and lifting instructions, avoiding the need to use additional languages, such as XSLT, to express the lowering and lifting scripts.

LOS and LIDS use SPARQL graph patterns to model inputs and outputs, thus providing a conjunctive logical description similar to our models. Therefore, all three approaches can model the relationships between inputs and outputs. However in LOS and LIDS, the graph patterns are represented as strings in the service description, limiting the power of automatic discovery and composition. In contrast, Karma uses RDF graphs to model the inputs and outputs, making it possible to discover services using SPARQL queries.

One clever feature of the LIDS approach is that each service invocation has a URI that embeds the input values and this URI is linked to the input resource. Sending a GET request to the invocation URI returns the linked output. This enables the user to get the RDF output on the fly while traversing the input resource on the Linked Data cloud without the requirement to send the input data in a separate phase. Although we have not implemented this capability, generating these kinds of URIs can be done in our approach without difficulty. We can use our service descriptions to find the appropriate input values in a triple store, create the invocation URIs, and link them to the input resources.

Verborgh et al. [16] introduce a new approach, called RESTdesc, to capture the functionality of hypermedia links in order to integrate Web APIs, REST infrastructure, and the Linked Data. The idea is to enable intelligent agents to get additional resources at runtime from the functional description of the invoked API. RESTdesc uses N3 notation to express the service description. Therefore, like LIDS, LOS, and our RDF vocabulary, it can model the relationships between the input and output attributes. However, the main point that differentiates our approach from RESTdesc is that in Karma, the user interactively builds API models. After modeling, Karma automatically generates API specifications and the API modeler does not need to write the descriptions manually.

7 Evaluation

To evaluate our approach, we modeled 11 Web APIs from the GeoNames Web services as linked APIs. The purpose of the evaluation was to measure the effort required in our approach to create linked APIs. We measured this effort in terms of the average time to build the linked API model and the number of action that the user had to perform to build the correct model. We used an extended version of the GeoNames ontology, with additional classes and properties to enable us to build richer semantic models.[14]

Table 1 shows the results of our experiment. The #URLs column indicates the number of sample invocation requests given to Karma by the user as the input of the modeling process. The #Cols column counts the number of columns in the Karma worksheet that were assigned a semantic type. The Choose Type column shows the number of times that the correct semantic type was not in Karma's top four suggestions and we had to browse the ontology to select the correct type. We started this evaluation with no training data. The Change Link column shows the number of times we had to select alternative relationships using a menu. The Time column records the time that it took us to build the model, from the moment the user enters the examples until the time that Karma publishes the service description into the repository.

As the results show, using Karma it took us only 42 minutes to build the models and deploy the linked APIs for the 11 Web APIs. In addition, we did not have to write any RDF, SPARQL, XPath or any other code. The whole process is done in a visual user interface, requiring 76 user actions, about 7 per Web API. Equivalent LIDS or LOS models are about one page of RDF/SPARQL/XSL code written by hand. We designed Karma to enable users to build these models quickly and easily enabling them to use the ontologies that make sense for their

[14] The datasets are available at: `http://isi.edu/integration/karma/data/iswc2012`

Table 1. Evaluation results for building linked APIs from the GeoNames APIs

GeoNames API	**#URLs**	**#Cols**	**#User Actions**			**Time(min)**
			Choose Type	**Change Link**	**Total**	
neighbourhood	3	10	10	6	16	6
neighbours	2	9	5	5	10	5
children	2	10	0	5	5	3
sibling	1	9	0	5	5	3
ocean	2	3	0	2	2	1
findNearby	3	11	0	5	5	3
findNearbyPostalCodes	3	11	1	5	6	7
findNearbyPOIsOSM	3	7	5	1	6	3
findNearestAddress	3	14	4	6	10	6
findNearestIntersectionOSM	3	8	5	3	8	3
postalCodeCountryInfo	1	5	3	0	3	2
Total	26	97	76			42

scenarios. Karma is available as open source[15] and we plan to collect usage statistics and feedback to improve the system and measure its benefits.

8 Discussion

This paper presented our approach to rapidly integrate the traditional Web APIs into the Linked Data cloud. An API modeler uses Karma to interactively build semantic models for the APIs. The system semi-automatically generates these models from example API invocation URLs and provides an easy-to-use interface to adjust the generated models. Our models are expressed in an RDF vocabulary that captures both the syntax and the semantics of the API. They can be stored in a model repository and accessed through a SPARQL interface. We deploy the linked APIs on a Web server that enables clients to invoke the APIs with RDF input and to get back the linked RDF data.

We are working to apply our modeling approach to a large number of available Web APIs. We plan to reduce the role of the user in modeling by mining the Web for examples of service invocations in documentation pages, blogs and forums to automatically construct datasets of sample data to invoke services. We are also working on extending our approach to model RESTful APIs. Extracting the input parameters from a RESTful API is not as straightforward as a Web API because there is not a standard pattern to embed the input values in the URL. However, collecting more samples of the API requests and analyzing the variable parts of the URLs will enable us to automatically extract the inputs of RESTful APIs.

Acknowledgements. This research is based upon work supported in part by the National Science Foundation under award number IIS-1117913. 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.

[15] `https://github.com/InformationIntegrationGroup/Web-Karma-Public`

References

1. Farrell, J., Lausen, H.: Semantic Annotations for WSDL and XML Schema, W3C Recommendation (2007), http://www.w3.org/TR/sawsdl/
2. Goel, A., Knoblock, C.A., Lerman, K.: Exploiting Structure within Data for Accurate Labeling Using Conditional Random Fields. In: Proceedings of the 14th International Conference on Artificial Intelligence, ICAI (2012)
3. Knoblock, C.A., Szekely, P., Ambite, J.L., Goel, A., Gupta, S., Lerman, K., Muslea, M., Taheriyan, M., Mallick, P.: 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)
4. Kou, L., Markowsky, G., Berman, L.: A Fast Algorithm for Steiner Trees. Acta Informatica 15, 141–145 (1981)
5. 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)
6. 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)
7. Lenzerini, M.: Data Integration: A Theoretical Perspective. In: Proceedings of the 21st ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS (2002)
8. Maleshkova, M., Pedrinaci, C., Domingue, J.: Investigating Web APIs on the World Wide Web. In: Proceedings of the 8th IEEE European Conference on Web Services, ECOWS (2010)
9. Norton, B., Krummenacher, R.: Consuming Dynamic Linked Data. In: First International Workshop on Consuming Linked Data (2010)
10. Pedrinaci, C., Domingue, J.: Toward the Next Wave of Services: Linked Services for the Web of Data. Journal of Universal Computer Science 16(13) (2010)
11. Pedrinaci, C., Liu, D., Maleshkova, M., Lambert, D., Kopecky, J., Domingue, J.: iServe: A Linked Services Publishing Platform. In: Proceedings of the Ontology Repositories and Editors for the Semantic Web Workshop, ORES (2010)
12. Speiser, S., Harth, A.: Towards Linked Data Services. In: Proceedings of the 9th International Semantic Web Conference (ISWC) (2010)
13. Speiser, S., Harth, A.: Integrating Linked Data and Services with Linked Data Services. 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. 170–184. Springer, Heidelberg (2011)
14. Taheriyan, M., Knoblock, C.A., Szekely, P., Ambite, J.L.: Semi-Automatically Modeling Web APIs to Create Linked APIs. In: Proceedings of the Linked APIs for the Semantic Web Workshop, LAPIS (2012)
15. Tuchinda, R., Knoblock, C.A., Szekely, P.: Building Mashups by Demonstration. ACM Transactions on the Web (TWEB) 5(3) (2011)
16. Verborgh, R., Steiner, T., Van Deursen, D., Coppens, S., Gabarró Vallés, J., Van de Walle, R.: Functional Descriptions as the Bridge between Hypermedia APIs and the Semantic Web. In: Proceedings of the 3rd International Workshop on RESTful Design (2012)

An Evidence-Based Verification Approach to Extract Entities and Relations for Knowledge Base Population

Naimdjon Takhirov[1], Fabien Duchateau[2], and Trond Aalberg[1]

[1] Norwegian University of Science and Technology, NO-7491 Trondheim, Norway
{takhirov,trondaal}@idi.ntnu.no
[2] Université Lyon 1, LIRIS, UMR5205, Lyon, France
fduchate@liris.cnrs.fr

Abstract. This paper presents an approach to automatically extract entities and relationships from textual documents. The main goal is to populate a knowledge base that hosts this structured information about domain entities. The extracted entities and their expected relationships are verified using two evidence based techniques: classification and linking. This last process also enables the linking of our knowledge base to other sources which are part of the Linked Open Data cloud. We demonstrate the benefit of our approach through series of experiments with real-world datasets.

Keywords: Linked Data, Knowledge Extraction, Machine Learning.

1 Introduction

The Web, which includes databases, catalogs and all textual documents, is a wealthy and a primary source of information. Thus, there is a need for exploiting this tremendous growth of the amount of online information as a source of structured knowledge [22]. Unfortunately, computers are not able to interpret this information due to a lack of semantics. However, the emergence of knowledge bases such as DBpedia and Freebase[1] in the Linked Open Data cloud (LOD), nowadays contain billions of facts expressed as RDF triples representing instances of relations between entities [4]. Researchers from various domains are increasingly interested in making their data available as part of the LOD, because a proper semantic integration of this data enables advanced semantic services. Examples of such services include exploratory search, supporting sophisticated and semantically rich queries, interoperability, question answering, etc. Converting the unstructured information, mainly the textual documents, to semantic models is therefore crucial to reach the expected Web of Data [15]. For instance, one of the most widely spread data representation used in the cultural heritage domain is *MARC* and its alternative forms. However, the Functional

[1] The complete list of interconnected bases can be found at http://linkeddata.org/

Requirements for Bibliographic Records (FRBR) model has gained much attention during the last decade as an underlying and much needed semantic data model for the cultural heritage data [20]. In this context, one of the most significant challenges deals with the **extraction of semantic information** hidden in plain documents [6,8,14]. Indeed, the textual documents are interesting because they may contain information that is otherwise missing or incomplete in the existing knowledge bases in the LOD cloud. Sentences in the documents include named entities which are connected with a specific type of relationship, e.g. *Martin Scorsese directed the movie The Departed*. Besides, the **interconnection of the LOD data sources** brings benefit for sharing and inferring knowledge [12]. Thus, extracting related entities from documents is not sufficient, and they need to be connected to the LOD cloud.

In this paper, we propose to tackle these two challenges. Our approach, KIEV[2] first extracts examples for a given relationship from textual documents. Indeed, some relationships are rarely encompassed in the structured data sources, but they can be found in textual documents (such as the Web). Mining these relationships with a pattern-based technique involves the discovery of a large amount of examples. Thus, a verification of these examples is performed at two levels: (i) the type of relationship is checked with a machine learning approach and (ii) the extracted entities are matched to LOD for both verification and integration purposes. In addition to these challenges, our approach KIEV should perform reasonably well in terms of efficiency at the Web scale since every page is a potential source of examples and good patterns. As a summary, the contributions of this paper are the following:

– We designed a generic approach for extracting semantic relationships from a large text corpora which integrates a verification process;
– These relationships are filtered and verified with a classification technique and an entity matching process. In addition, the link from our generated entity to its corresponding LOD entity enables the connection and possible reasoning over all interconnected knowledges bases;
– Finally, we have conducted experiments with real-world datasets (about movies and sports) to evaluate the quality and the robustness of our approach.

The rest of this paper is organized as follows. Section 2 introduces the formalization of our problem and provides an overview of KIEV. Section 3 covers the first part of our approach, the discovery of examples by using patterns, while Section 4 and 5 focus on the evidence-based verification of these examples. The related work is described in Section 6. Our experiments are detailed in Section 7. Finally, we conclude in Section 8.

2 Overview of our Approach

Our goal can be seen as **the creation of a knowledge base of entities and relationships**. Simply assuming the existence of a repository of domain

[2] KIEV – **K**nowledge and **I**nformation **E**xtraction with **V**erification.

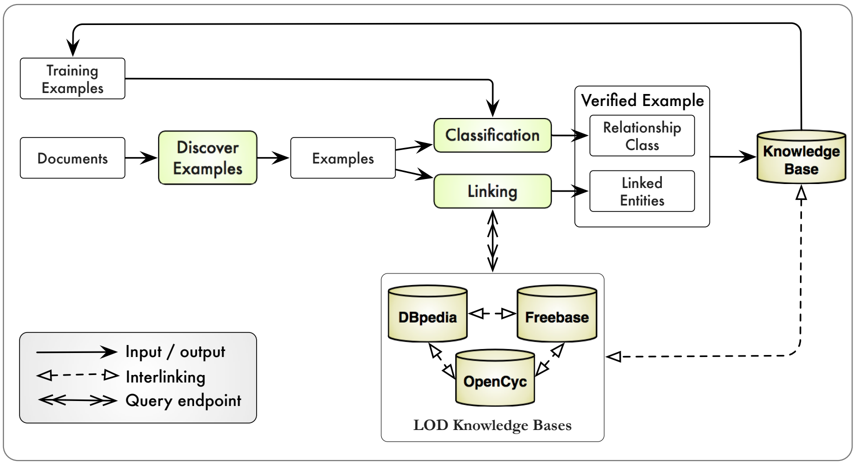


Fig. 1. Overview of our Approach

entities would limit our approach. Rather, we extract entities from the textual documents, and as a consequence, our approach should also work with entities which have been previously identified (i.e., from a repository). A **relationship** is defined as a triple <*entity*$_1$, *type-of-relationship, entity*$_2$>. As an example, considering the 2006 *"The Departed"* movie directed by Martin Scorsese as a remake of the Andrew Lau's *"Infernal Affairs"* from 2002, the example would be represented as <*"Infernal Affairs", hasImitation, "The Departed"*>.

Figure 1 depicts the global overview of KIEV. Given a type of relationship, KIEV requires a collection of documents and a few training examples (verifying the types of relationship) to bootstrap a possible infinite loop. The first step consists of **discovering examples** from the textual collection (see Section 3). It is based on semantic tagging which combines Named Entity Recognition and Part of Speech tagging, and it generates many examples for the concepts contained in a sentence. Thus, a verification of the relevance for these examples is performed with two other processes. The former checks if the extracted entities are effectively related with the type of relationship using a **machine learning classifier** (see Section 4). The latter process **links both extracted entities** of an example to their corresponding entities on the LOD cloud (see Section 5). Once an example is verified, it can be used as a training example to improve the classifier, but also to reinforce the confidence score of a pattern during the discovery process.

3 Discovering Examples

The core idea of our approach is to process the input as a stream of documents and to iteratively update our semantic knowledge base of entities. In this section, we describe the first part of our approach – discovering examples. An example for

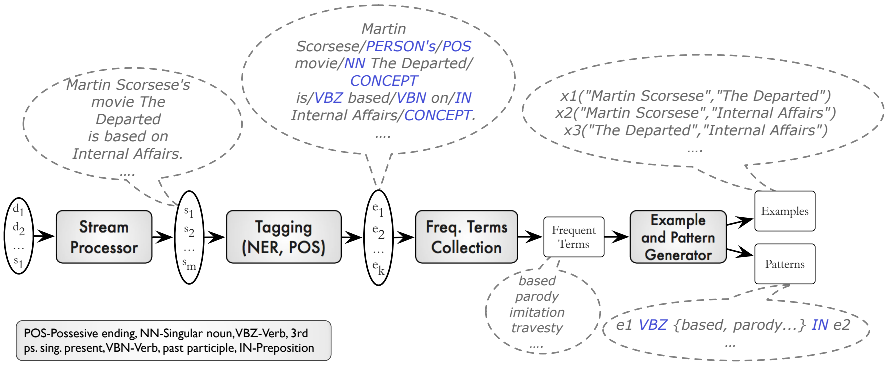

Martin Scorsese's movie The Departed is based on Internal Affairs. ....
Martin Scorsese/PERSON's/POS movie/NN The Departed/CONCEPT is/VBZ based/VBN on/IN Internal Affairs/CONCEPT. ....
x1("Martin Scorsese","The Departed")
x2("Martin Scorsese","Internal Affairs")
x3("The Departed","Internal Affairs")
....
d1 d2 ... s1
Stream Processor
s1 s2 ... sm
Tagging (NER, POS)
e1 e2 ... ek
Freq. Terms Collection
Frequent Terms
Example and Pattern Generator
Examples
Patterns
based parody imitation travesty ....
e1 VBZ {based, parody...} IN e2 ...
POS-Possesive ending, NN-Singular noun, VBZ-Verb, 3rd ps. sing. present, VBN-Verb, past participle, IN-Preposition

Fig. 2. Workflow of Processes for Discovering Examples

a given type of relationship is composed of two entities (e.g., for *imitation* type of relationship, an example is <*"Infernal Affairs", "The Departed"*>). Figure 2 provides the big picture of the example discovery workflow, whose goal is to generate a set of examples. Each process in the workflow of discovering examples is presented below.

3.1 Stream Processing

Stream processor (SP) accepts as an input documents in textual form. The first task the SP performs is to pre-process the input. For example, this task may involve cleaning the html documents for tags, removing headers from emails, etc . At this point, we are interested in only obtaining text regardless of the quality. Each document $d \in \mathcal{D}$ is segmented into a list of sentences such that $d = \{S_i \mid i = 1 \ldots N\}$ where N is the number of sentences. A sentence S_i is discarded if S_{i-1} and S_{i+1} contain no entities. This is because S_i may contain a personal pronoun referring to the previous sentence, e.g. *"**Martin Scorsese** is an American film director. **He** is the creator of Taxi Driver."*. Additionally, the sentences are filtered out to eliminate those that were likely to be noisy (broken and invalid sentences) and not useful for example discovery (e.g., non-English sentences, sentences missing verb, sentences with only uppercase letters or only with lowercase letters, sentences without capital letters, etc.). The next step deals with the semantic tagging of the selected sentences with named entity recognition (NER) and part-of-speech (POS) tags.

3.2 Tagging

For each sentence $s \in S_i$, named entity recognition is performed to detect the set of entities $\mathcal{E}$ (person, location, organization, dates, etc.). Consider a document containing the following sentence: *Infernal Affairs was followed by a 2006 American remake by Martin Scorsese entitled The Departed.* From this sentence, two *concepts* are detected and one *person*. Traditionally, NER is focused around the detection of common entities such as people, organization or geographic location. However, the recognition of domain specific entities poses a particular challenge because the NER tools usually require training examples for the types of entities to recognize. In our context of textual documents from the Web, providing such examples is not possible.

To avoid missing entities, a POStagger is first applied on all sentences. Our assumption is that entities are POStagged as *"noun"*. Thus, we consider that **all nouns in the sentences are entities**. A NER tool can confirm some of these entities. Although this assumption implies the identification of many incorrect entities, the next steps are in charge of discarding those irrelevant entities. The output of the semantic tagger is a set of semantically and structurally tagged sentences, from which we can extract frequent terms.

3.3 Frequent Terms Collection

Terms that appear frequently in the same sentence with a pair of entities are likely to be **highly relevant to the pair of entities**. For example, in the sentence *"Martin Scorsese's movie The Departed is based on Internal Affairs"*, frequent terms are *movie* and *based on* because they appear frequently together with the entities in the sentence.

In order to collect these frequent terms, all possible word n-grams are first identified in the sentence s. The top thousand most common words on the Web[3] are excluded and cannot be part of frequent terms. Then, the sentence s is splitted into a set of words. A list of n-grams is constructed out of this list. After the list of n-grams has been obtained, we look up Wordnet lexical database to obtain the list Φ of semantically related words. These words are grouped into unordered sets (synsets). Stopwords (e.g., "the", "a", "but" etc.) are removed and stemming is performed. The following Wordnet relations are used:

- synonymy (e.g., "writer" and "novelist"), words that denote the same concept and are interchangeable in many contexts.
- hyponym, a word whose semantics are included within that of another word[4], e.g., "The Departed is a movie".

Since the synsets obtained from Wordnet have a shared information content, i.e., hierarchy of is-a concepts, this list of semantically similar words can be larger than desired. Thus, to control the level of granularity of this list of concepts, we employ the Resnik similarity to prune those that are below a given threshold [16]. This similarity measure is applied between the segmented n-grams and each of the synsets in Φ. For example, the distance between "novel" and "book" is 0.29.

These frequent terms are generated for different objectives such as the classification of examples through features, but also to generate the examples as explained in the next part.

[3] This list is available from Microsoft Web N-gram Service: `http://bit.ly/bFKSxz`
[4] This is similar to *is-a* relationship.

3.4 Example and Pattern Generator

Having obtained the lists of named entities and frequent terms, a **set of candidate examples** is built. One of our goals is to populate a knowledge base that can serve as a repository of distinct entities. First, a set of unique pair of entities Θ is constructed such that $\Theta = \{(e_i, e_j) | e_i \neq e_j, e_i \in \mathcal{E}, e_j \in \mathcal{E}\}$. At first glance, it appears that we generate overly many examples and this most likely leads to a fair number of false positives. But we will show in section 4 that our classification approach effectively discards these irrelevant examples. Our basic assumption with generating so many examples is to reduce the likelihood of low recall.

At this time, we can **generate patterns** based on the information from the frequent terms collector. That is, we mask the named entities (e.g. "Infernal Affairs" $\Rightarrow e_1$, "The Departed" $\Rightarrow e_2$). The idea is to obtain entity and word independent patterns, as shown in the Figure 2. At the end of each iteration, a list of patterns is generated from the candidate examples. If the pattern had been generated before, its statistics are updated from the current iteration. For patterns $\{p_1, \ldots, p_n\}$, we compute the pattern similarity using the Levenshtein distance and those above a given threshold are merged into a set of patterns P_p. By now, we know the amount of patterns generated in this iteration (P_i). We note the list X_p of examples that support this pattern. The patterns generated at iteration i are ranked according to the following scoring function:

$$score(p) = \frac{\alpha \, \frac{occ(p)}{i} + \beta \frac{|P_p|}{|P_i|} + \gamma \frac{|X_p|}{|X|}}{\alpha + \beta + \gamma}$$

where $occ(p)$ is the number of iterations this pattern has been discovered out of total number of iterations i. X denotes the number of total examples in the system. The scores are normalized in the range $[0, 1]$. The patterns generated during this iteration will be used to discover new examples in the next iteration. These patterns will also be used as features during the classification process.

As previously explained, all of the examples discovered so far may not be correct. In the next section, we will show how a classifier effectively discards false positives.

4 Classification

The first part of the verification is to check that the candidate entities (represented with a label) are related with a type of relationship. Indeed, a sentence may contain different entities and the discovery process generates in that case incorrect examples, mainly because of the pattern-based matching. The classification aims at discarding these incorrect examples without prior knowledge about the two entities. To fulfill this goal, the verification process can be seen as a **classification problem** [13]. Given a set of features (properties), the idea is to find the correct class for a given example (extracted from a sentence). Each class represents a type of relationship (e.g., *imitation, adaptation*). For instance, the example *(James Cameron, Avatar)* should be classified in the class *creatorOf*. A specific class named *unknown relationship* is added as a garbage class to collect all incorrect examples or those that cannot be classified in another class. To select the correct class for an example, a classifier is trained using training examples, i.e., examples for which the correct class is already known. Although the training process depends on the type of classifier (e.g., decision tree, Bayes network), it mainly consists of minimizing the misclassification rate when classifying the training examples according to the values of their features [13]. To compute these values, each training example is used as a query over the document collection and all sentences containing the two entities of the example are analyzed given the following features: the frequency and the presence of any frequent terms (e.g., *parody*), the length and structure of the best-ranked pattern which generated the example (see Section 3.4), the average spamscore of the documents from which the pattern is extracted [7]. Note that this paper does not aim at designing a new classifier, but we rather use existing ones from the Weka environment [9]. More formally, an example $x \in \mathcal{X}$ is defined by a set of features $\mathcal{F}$. We note the set of training examples $\mathcal{T}$, with $\mathcal{T} \subseteq \mathcal{X}$. Each example can be assigned a class $c \in \mathcal{C}$. Given a (type of) classifier $\mathit{\Gamma}$, we formulate the training as a process to obtain an instance γ of this classifier as follows:

$$\mathit{\Gamma}(\mathcal{T}, \mathcal{F}, \mathcal{C}) \rightarrow \gamma$$

The advantage of building a generic classifier rather than many binary classifiers (for each type of relationship) is that the former enables the verification of different types of relationships. Consider a query for *"imitation"*, we could obtain the pair of entities <*"Infernal Affairs", "The Departed"*> and <*"The Departed", Martin Scorsese"*>. With a binary classifier for "imitation", we would only keep the first example. With a generic classifier, we would store both examples (classified in different classes). When an instance of a classifier which best minimizes the misclassification rate is trained, we can use this instance γ for assigning classes to the unclassified examples:

$$\gamma(\mathcal{X}, \mathcal{F}, \mathcal{C}) \rightarrow <(x_1, c_1), (x_2, c_1), (x_3, c_4), \ldots, (x_k, c_n)>$$

In our context, we cannot assume that the user provides many initial training data. A set of 5 to 10 examples for each class is realistic. However, some classifiers are robust with a few training examples while other classifiers achieve better results with more training data. Two problems arise from these remarks: the former is about selecting which examples should be added as training data while the latter deals with the choice of the classifier for each iteration. Let us discuss **the choice of the training data** first. To improve the robustness of the classifier, one has to train it with more data. To add new examples as training data, we have to select them among the sets of discovered examples from the previous iterations. We propose two strategies to achieve this goal. The first one (linking based) consists in selecting all examples that have been verified (with the classification step and the linking process) during any previous iterations. The second strategy (frequency based) is based on a frequency constraint: all examples which have been discovered in half of the previous iterations are added as training data during the current iteration. We believe that this selection of training data could be investigated further, e.g., when combining the two described strategies.

As for **the selection of the classifier**, the idea is the following: with the selected training examples, we generate instances of different types of classifiers (decision trees such as *J48* or *NBTree*, instance-based such as *KStar* or *IBk*, rule-based such as *NNge* or *JRip*, etc.). We perform cross-validation against the set of training examples for each instance of a classifier, and we compute the misclassification rate for each of them. The instance of classifier which achieves the minimal misclassification rate is selected to classify the examples discovered at this iteration. Such a strategy enables us to ensure that the best classifier is used for each iteration, but it also brings more flexibility to our approach.

We will show the impact of the training data and the type of classifiers in Section 7. The result of the classifier is a set of pairs, each of them composed of an example and its verified relationship class. The next step is to check whether the two extracted entities have a corresponding LOD entity.

5 Entity Linking

Entity Linking is the task of discovering local entity's correspondence in another data source [19]. The interest in linking entities is increasing rapidly due to the LOD movement. Note that linking does not imply coreference resolution is performed, but linking partially solves the coreference resolution problem. For example, the local entities "Martin Scorcese" and "Scorcese" are both linked to the same DBpedia entity *Martin_ Scorsese*. The kind of linking we are performing here differs from structure-based linking as we **only have labels** at our disposal. The core of the idea is to **match the entity against existing general purpose semantic knowledge bases** such as DBpedia or Freebase to obtain corresponding LOD entities. Namely, we build various queries by decomposing the initial label and we query in the *descriptive text* attributes of knowledge bases (i.e., *common.topic.article* for Freebase, *dbpedia-owl:abstract* for DBpedia, etc.). In most cases, several candidate entities are returned and the task deals with automatically selecting the correct one. To fulfill this goal, the intuition is based on the hypothesis that the document about entity e and the descriptive text of LOD entity l should be fairly similar. Linking is performed for each entity of each document. That means that each document where e is mentioned serves as a context for disambiguation and matching against LOD knowledge bases. We note $\boldsymbol{\xi}$ the vector of terms in e's document, while $\boldsymbol{\Lambda}$ represents the vector of terms of l. Terms in both documents are treated using *bag-of-words* method and both the context of e and the descriptive text of l are represented as a point in an n-dimensional term space. The cosine similarity score between the vectors $\boldsymbol{\xi}$ and $\boldsymbol{\Lambda}$ is calculated as follows:

$$sim(\boldsymbol{\xi}, \boldsymbol{\Lambda}) = \frac{\sum_{i=1}^{n} \boldsymbol{\xi}_i \times \boldsymbol{\Lambda}_i}{\sqrt{\sum_{i=1}^{n} (\boldsymbol{\xi}_i)^2} \times \sqrt{\sum_{i=1}^{n} (\boldsymbol{\Lambda}_i)^2}}$$

where n is the size of the vocabulary. The terms in both vectors are based on classical *tf/idf* scores while the vocabulary is created out of the whole document collection. The top ranked entities are chosen as candidates for further comparison. This last comparison is performed on labels (and optionally "redirects" property) of the two entities to ensure a reasonable similarity in the label of e and one of the labels of l (e.g. "rdfs:label" and "dbpedia-owl:wikiPageRedirects" for DBpedia). This comparison is necessary because even though the similarity function returns a sufficiently high cosine similarity score, the labels should also be lexically similar. At this stage, the three well-known similarity measures are applied (Jaro Winkler, Monge Elkan and Scaled Levenshtein) as described in [19]. The top linked LOD entity is stored and is considered as a candidate until the end of the iteration. At the end of an iteration, all verified relationships (both by the classification and the linking) are **converted into triples**: for each entity, some triples express the link to LOD, the different labels and other possible attributes. One triple represents the relationship between the two entities and the type of relationship. Thus, the knowledge base is populated iteratively and can run continuously.

6 Related Work

In the field of knowledge extraction, various works have been proposed to discover relationships for specific domains [22]. For instance, **Snowball** associates companies to the cities where their headquarter is located [1] while **DIPRE** focuses on books and authors [5]. To increase the quality and the consistency of generated facts, systems may either be based on general ontologies such as Yago [14] or on logical rules associated with a SAT solver [18]. The last trend in this domain deals with **Open Information Extraction**, in which the large scale aspect of the Web is taken into account [8]. However, none of these works clearly aim at building a semantic knowledge base, thus there is no linking with the LOD cloud.

DBpedia is one of the first initiative to automatically extract structured content from Wikipedia [4]. It relies on the infoboxes provided by the knowledge-sharing community. Since, many companies and organizations have added their own knowledge base to the LOD cloud, from generic ontologies such as **Yago** and **Freebase** to specialized bases such as **MusicBrainz** or **LinkedMDB** [10]. The process for converting unstructured or semi-structured data sources into facts is called *Triplification*[5]. For instance, **Triplify** has been designed to extract triples from Relational databases and expose them on LOD [2] while **Catriple** builds a store of triples from Wikipedia categories [11]. Similarly to most of these approaches, we generate triples and store them in our knowledge base.

In the Information Retrieval domain, researchers have studied the discovery of the corresponding LOD entities for a given task, such as in the **TREC challenge** [3]. Due to the large scale application and the uncertainty of the results, a ranking of the most probable entities which correspond to the query (usually

[5] `http://triplify.org/Challenge/`

with target categories) is computed [21,17]. The linking to LOD for disambiguation and enrichment has also been studied for any bag of words [12] as well as for FRBR entities [19]. In our context, the entities are extracted from textual documents and usually represented with a label. The surrounding context of the label in the sentence is the main information available for discovering the corresponding LOD entity.

7 Experimental Evaluation

To assess the effectiveness of our approach, we have conducted a number of experiments which are presented below.

7.1 Experimental settings

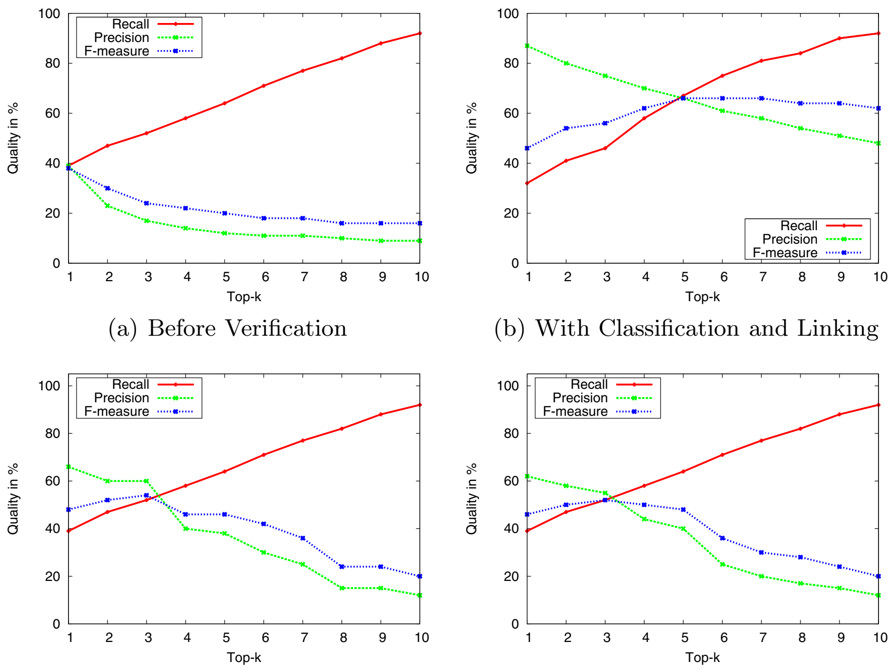
Our **document collection** is the English subset of the ClueWeb09 dataset[6] which consists of 500 million documents. This dataset is used by several tracks of the TREC conference [3]. For **semantic tagging**, several text processing tools have been used, including OpenNLP[7] (for tokenization and sentence splitting), the StanfordNLP[8](for POS tagging). For **classification**, six classifiers of different types were applied, namely the classic Naïve Bayes, the rule-based (NNge, DecisionTable), tree-based (J48, RandomForest) and lazy (KStar). These classifiers are included in the Weka software [9]. As for **linking**, we have used the DBpedia[9] dataset version 3.7 which contains 3,550,567 triples. Apache Lucene was employed for the **backend indexing**. Running KIEV for one type of relation on a subset of the collection took roughly 20 minutes.

7.2 Quality of Discovery Process

In this experiment, we focus on the movie dataset (remakes). Examples of relationships of interest include *imitation*, *adaptation* and *creator*. The ground truth for this dataset was obtained from the IMDb[10] movie database. This ground truth contained 545 entries out of the total 1052 remake pairs. For the remaining 507 we could not find suitable documents in our collection. The reason for this is twofold. First, a number of movies were in non-English language. Second, a significant number of movies were created before the *Information Age*, i.e., those produced earlier than 1970s. Additionally, some examples were only mentioned in a few documents.

Figure 3 demonstrates the results of our experiments **with or without the evidence-based verifications**. The quality is presented in terms of well-known

[6] http://lemurproject.org/clueweb09/
[7] http://opennlp.apache.org/
[8] http://nlp.stanford.edu/
[9] http://wiki.dbpedia.org/Downloads37
[10] http://imdb.com



(a) Before Verification (b) With Classification and Linking

(c) Linking only (d) Classification only

Fig. 3. Quality Results for the Remakes Dataset

information retrieval measures - recall, precision and F-measure. Extracted examples are ranked and thus presented by top-k. In our context, the recall (at top-k) is the fraction of extracted correct examples (at top-k) out of the total number of correct examples (at top-k), while the precision (at top-k) is the number of extracted correct examples (at top-k) out of the total number of extracted examples (at top-k). F-measure is the harmonic mean of precision and recall.

We first notice that before the verification process, the precision score is quite low (≈39%) at top-1. This is because the discovery process extracts quite a lot of incorrect examples (false positives). As we increase the top-k, the recall also increases and eventually peaks at 87% at top-10. This trend illustrates that our approach achieves fairly high recall value but at the expense of precision. We tackle this issue with our verification techniques, i.e., classification and linking. To show the **benefit of both verification steps**, the individual results of these steps are depicted in Figure 3(c) and 3(d). The recall value for both classification only and linking only is very similar to the values before verification, thus confirming that the individual verification do not discard many correct relationships. And the precision values for both steps, which are lower than the precision score after verification at any iteration, indicate that classification and linking do not discard the same incorrect examples. Thus, they enable a higher precision when they are combined.

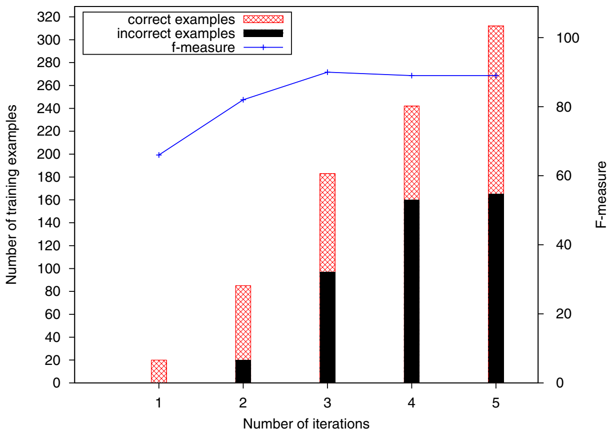


Fig. 4. Impact of Training Examples with Frequency based Strategy

Figure 3(b) illustrates that the **verification process** is effective to discard incorrect examples (precision score reaching ≈85% at top-1). However, a few correct relations were also discarded (a ≈6% decrease of recall at top-1), mainly due to the missing of a link to LOD of one of the entities. Furthermore, this phenomenon involves changes in the ranking of the extracted examples. Correct relationships can be promoted to a higher top, thus increasing the recall value of the highest top (e.g., at top-5). Finally, the benefit of the verification process clearly appears at top-10, since the plots have a close recall value (≈87%) but the verification discarded half of the incorrect examples (50% precision).

7.3 Impact of the Training Data

The example discovery process *feeds* the classifier with new training data for the subsequent iteration. In this experiment, we have studied the impact of the selection of this training data by comparing the two strategies described in Section 4.

Frequency Based Strategy. The frequency based strategy accounts for the frequency of a given example being discovered in all iterations. Initially, the user provides a set of 20 training examples (5 per relation type). If a given example is discovered repeatedly on each iteration, the intuition behind this strategy is that this example is most likely valuable and is promoted as a training example in the next iteration. Figure 4 illustrates the impact of the training data at the i-th iteration. On the y axis, we have the number of training examples that is used by our classifiers. On the right $y2$ axis, we have the harmonic mean F-measure obtained by the best performing classifier at the i-th iteration. The best performing classifier is the one with the highest F-measure during the classification with

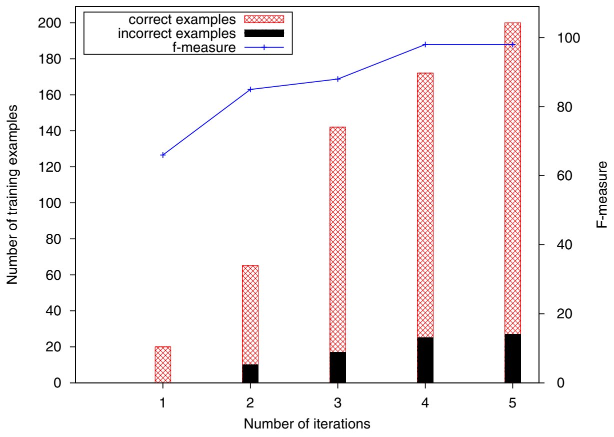


Fig. 5. Impact of Training Examples with Linking based Strategy

10-fold cross-validation against the training data. Note that from one iteration to the other, the best performing classifier may be different because the set of training data evolves. For example, *KStar* was selected as the best classifier for the first iteration, but *J48* performed better in the second iteration.

On the plot, 85 examples discovered during the first iteration are selected for training for the second iteration. However, 20 of them are incorrect, i.e. false positives (shown as a black bar). Both the number of correct and incorrect examples increases as we move towards the 5-th iteration, eventually reaching 312 and 165 examples respectively for correct and incorrect examples. The high number of examples can be explained as follows. The frequency based strategy promotes as training data examples which appear at least 50% of the time in the previous iterations. Thus the number of added examples can potentially grow high. Yet, the F-measure obtained on the remakes dataset does not suffer much from the presence of incorrect examples (stable around 89% after the 3-rd iteration).

Linking Based Strategy. The linking based strategy provides a harder constraint than the frequency based strategy when selecting the training data. Indeed, the candidate examples have to be verified both by the classification and by the linking process. Let us study the impact of this strategy over the quality of results by analyzing Figure 5. It presents the F-measure value achieved by the best generated classifier and the evolution of the number of training examples for five iterations.

The first remark about this plot deals with the F-measure scores, which are higher than those of the frequency based strategy from iterations 1 to 5. Another interesting phenomenon with this strategy is that the number of examples selected as training data (y axis) is lower than the one of the frequency based strategy. Indeed, the linking based strategy requires that both entities of an example are linked to LOD. Thus, this number is dramatically reduced, i.e., 200 in linking based strategy versus 312 in the frequency-based strategy at the fifth iteration. Finally, the number of incorrect examples is much lower in the linking based strategy too.

These remarks about the total number of correct examples together with the higher F-measure value are clear indicators that the linking based strategy is quality oriented while the frequency based strategy is performance oriented (simple and fast computation). The latter strategy is more appropriate for quickly generating training examples.

7.4 Comparative Evaluation

Finally, the evaluation of KIEV would not be complete without a **comparison with similar knowledge extraction systems**. Two systems, Prospera and NELL, are publicly available along with their dataset about *sports*. The results of these systems over the *sport* dataset are reported in [6,14]. To be fair in this evaluation, we have used the same set of training examples, and we also validated 1000 random types of relationship, as explained in the experiments reported in [6,14]. This means that similarly to Prospera and NELL, our precision is an estimation, due to the amount of relationships to be validated.

Figure 6 summarizes the comparison between the three systems in terms of estimated precision. We notice that the average precision of the three systems is the same (around 0.91). However, the total number of facts discovered by KIEV $(71,921)$ is 36 times higher than NELL $(2,112)$ and 1.3 times higher than Prospera $(57,070)$. As a consequence, KIEV outperforms both baselines. Prospera provides slightly better quality results than our approach on the *AthletePlaysForTeam* relationship. However, several factors have an influence on the precision results between Prospera, NELL and KIEV. First, Prospera is able to use seeds and counter seeds while we only rely on positive examples. On the

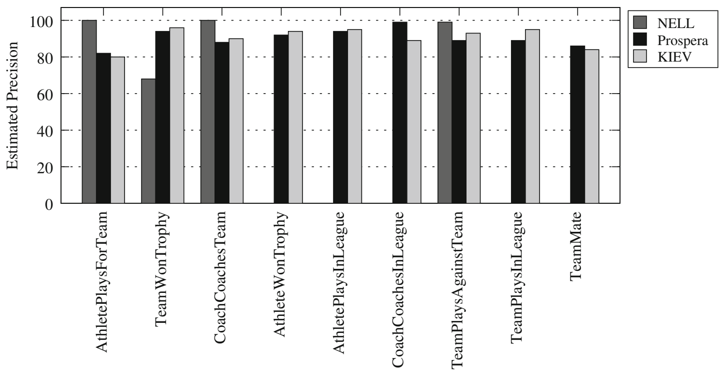


Fig. 6. Comparison to NELL and Prospera

other side, Prospera includes a rule-based reasoner combined with the YAGO ontology and KIEV mainly uses the LOD cloud for verification purposes. Yet, the combination of POS-tagged patterns and NER techniques supported by the two verification steps achieves outstanding precision values.

8 Conclusion

We have presented our novel approach KIEV for **populating a knowledge base** with entities and relationships. Our approach enables the analysis of a large amount of documents to extract examples (of entities) with their expected type of relationships after each iteration. A **verification step** ensures an acceptable quality for these extracted relationships by discarding irrelevant examples (classification) and by discovering the corresponding LOD entities (entity linking). Experiments performed on different datasets confirm the significant benefit of the verification step, thus enabling our approach to run continuously and to use new examples as training data to strengthen both the produced classifier and consequently the verification process.

The outcome of this work provides several interesting perspectives. First, we plan to run more experiments to **analyze the impact of parameters** (e.g., selection of the training data, number of iterations on the long term). We could associate a confidence score (based on provenance, number of patterns, number of occurrences, etc.) to each discovered relationship to rank them and help discarding the incorrect ones. Another objective is to study the **architecture and implementation of the knowledge base** in terms of infrastructure and support for RESTful and SPARQL queries. When our knowledge base will be publicly available, we plan to **integrate user feedback** to address the potentially contradictory cases between the two verification steps (classification and linking). Then, an extension could be proposed to **discover any type of relationship**, from an ontology for instance, by automatically defining the features and the training examples.

References

1. Agichtein, E., Gravano, L.: Snowball: Extracting Relations from Large Plain-Text Collections. In: Proceedings of ACM DL, pp. 85–94 (2000)
2. Auer, S., Dietzold, S., Lehmann, J., Hellmann, S., Aumueller, D.: Triplify: light-weight linked data publication from relational databases. In: Proceedings of WWW, pp. 621–630 (2009)
3. Balog, K., Serdyukov, P., de Vries, A.P.: Overview of the TREC 2011 entity track. In: Proceedings of TREC 2011. NIST (2012)
4. Bizer, C., Heath, T., Berners-Lee, T.: Linked Data - The Story So Far. International Journal of Semantic Web and Information Systems 5(3), 1–22 (2009)
5. Brin, S.: Extracting Patterns and Relations from the World Wide Web. In: Atzeni, P., Mendelzon, A.O., Mecca, G. (eds.) WebDB 1998. LNCS, vol. 1590, pp. 172–183. Springer, Heidelberg (1999)

6. Carlson, A., Betteridge, J., Kisiel, B., Settles, B., Hruschka Jr., E., Mitchell, T.M.: Toward an architecture for never-ending language learning. In: Proceedings of AAAI, pp. 1306–1313 (2010)
7. Cormack, G.V., Smucker, M.D., Clarke, C.L.A.: Efficient and effective spam filtering and re-ranking for large web datasets. Inf. Retr. 14(5), 441–465 (2011)
8. Etzioni, O., Fader, A., Christensen, J., Soderland, S., Mausam: Open Information Extraction: The Second Generation. In: Proceedings of IJCAI, pp. 3–10 (2011)
9. Garner, S.R.: WEKA: The Waikato Environment for Knowledge Analysis. In: Proceedings of the New Zealand Computer Science Research Students Conference, pp. 57–64 (1995)
10. Hassanzadeh, O., Consens, M.P.: Linked Movie Data Base. In: Proceedings of LDOW (2009)
11. Liu, Q., Xu, K., Zhang, L., Wang, H., Yu, Y., Pan, Y.: Catriple: Extracting Triples from Wikipedia Categories. In: Domingue, J., Anutariya, C. (eds.) ASWC 2008. LNCS, vol. 5367, pp. 330–344. Springer, Heidelberg (2008)
12. Milne, D., Witten, I.H.: Learning to link with wikipedia. In: Proceedings of CIKM, pp. 509–518 (2008)
13. Mitchell, T.: Machine Learning. McGraw-Hill Education (ISE Editions) (October 1997)
14. Nakashole, N., Theobald, M., Weikum, G.: Scalable knowledge harvesting with high precision and high recall. In: Proceedings of WSDM, pp. 227–236 (2011)
15. Parundekar, R., Knoblock, C.A., Ambite, J.L.: Linking and Building Ontologies of Linked Data. 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. 598–614. Springer, Heidelberg (2010)
16. 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 (JAIR) 11, 95–130 (1999)
17. Rode, H., Serdyukov, P., Hiemstra, D.: Combining document- and paragraph-based entity ranking. In: Proceedings of ACM SIGIR, pp. 851–852 (2008)
18. Suchanek, F., Sozio, M., Weikum, G.: SOFIE: a self-organizing framework for information extraction. In: Proceedings of WWW, pp. 631–640 (2009)
19. Takhirov, N., Duchateau, F., Aalberg, T.: Linking FRBR Entities to LOD through Semantic Matching. In: Gradmann, S., Borri, F., Meghini, C., Schuldt, H. (eds.) TPDL 2011. LNCS, vol. 6966, pp. 284–295. Springer, Heidelberg (2011)
20. The International Federation of Library Associations and Institutions. Functional requirements for bibliographic records. In: UBCIM Publications - New Series, vol. 19 (1998)
21. Vercoustre, A.-M., Thom, J.A., Pehcevski, J.: Entity ranking in Wikipedia. In: Proceedings of ACM SAC, pp. 1101–1106 (2008)
22. Weikum, G., Theobald, M.: From information to knowledge: harvesting entities and relationships from web sources. In: Proceedings of PODS, pp. 65–76 (2010)

Blank Node Matching and RDF/S Comparison Functions

Yannis Tzitzikas, Christina Lantzaki, and Dimitris Zeginis[⋆]

Computer Science Department, University of Crete,
Institute of Computer Science, FORTH-ICS, Greece
{tzitzik,kristi,zeginis}@ics.forth.gr

Abstract. In RDF, a *blank node* (or anonymous resource or bnode) is a node in an RDF graph which is not identified by a URI and is not a literal. Several RDF/S Knowledge Bases (KBs) rely heavily on blank nodes as they are convenient for representing complex attributes or resources whose identity is unknown but their attributes (either literals or associations with other resources) are known. In this paper we show how we can exploit blank nodes anonymity in order to reduce the delta (diff) size when comparing such KBs. The main idea of the proposed method is to build a mapping between the bnodes of the compared KBs for reducing the delta size. We prove that finding the optimal mapping is NP-Hard in the general case, and polynomial in case there are not directly connected bnodes. Subsequently we present various polynomial algorithms returning approximate solutions for the general case.

For making the application of our method feasible also to large KBs we present a signature-based mapping algorithm with $n \log n$ complexity. Finally, we report experimental results over real and synthetic datasets that demonstrate significant reductions in the sizes of the computed deltas. For the proposed algorithms we also provide comparative results regarding delta reduction, equivalence detection and time efficiency.

1 Introduction

The ability to compute the differences that exist between two RDF/S Knowledge Bases (KBs) is an important step to cope with the evolving nature of the Semantic Web (SW). In particular, RDF/S Deltas can be employed to aid humans understand the evolution of knowledge, and to reduce the amount of data that need to be exchanged and managed over the network in order to build SW synchronization [19,1], versioning [7,8,1,4,21] and replication [17] services. A rather peculiar but quite flexible feature of RDF is that it allows the representation of *unnamed* nodes, else called *blank nodes* (for short bnodes), a feature that is convenient for representing complex attributes (e.g. an attribute `address` as shown in Figure 1) without having to name explicitly the auxiliary node that is used

[⋆] Current affiliation: Information Systems Lab,University of Macedonia, Thessaloniki, Greece, `zeginis@uom.gr`.

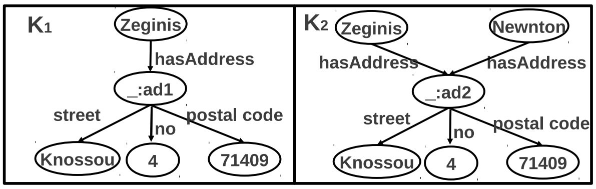


Fig. 1. Examples of blank nodes

for connecting together the values that constitute the complex value (i.e. the particular `street, number` and `postal code` values). A recent paper [10] that surveys the treatment of bnodes in RDF data, proves that blank nodes is an inevitable reality. Just indicatively, and according to their results, the data fetched from the "hi5.com" domain consist of 87.5% of blank nodes, while those from the "opencalais.com" domain, which is part of LOD (Linked Open Data) cloud, has 44.9% bnodes. The authors also state that the inability to match bnodes increases the delta size and does not assist in detecting the changes between subsequent versions of a KB.

Previous works on comparing RDF KBs have not elaborated on this issue thoroughly. There are works (e.g. [21,22]) proposing differential functions that yield reduced in size deltas (in certain cases) but treat bnodes as named nodes. Other works and systems (specifically Jena [3]) focus only on deciding whether two KBs that contain bnodes are equivalent or not, and do not offer any delta size saving for the case where the involved KBs are not equivalent. In brief, and to the best of our knowledge, there is not any work that attempts to establish a bnode mapping for reducing the delta size for the case of non equivalent KBs. Note that finding such a mapping can be considered as a preprocessing step, a task that is carried out before a differential function (like those described in [17,20,16,15,8,21]) is applied.

We prove that finding the optimal mapping is NP-Hard in the general case and polynomial if there are not directly connected bnodes. Subsequently we present various polynomial algorithms returning approximate solutions for the general case. For making the application of this method feasible also to large KBs one of these algorithms has $n \log n$ complexity.

The experimental results over real and synthetic datasets showed that our method significantly reduces the sizes of the computed deltas, while the time required is affordable (just indicatively the $n \log n$ algorithm requires a few seconds for KBs with up to 150,000 bnodes). For the proposed algorithms we also provide comparative results regarding time efficiency and their potential for delta reduction and equivalence detection.

The rest of this paper is organized as follows. Section 2 discusses RDF KBs with bnodes and the equivalence of such KBs. Section 3 elaborates on the problem of finding the optimal mapping. Section 4 proposes bnode matching algorithms and Section 5 reports experimental results. Section 6 discusses the applicability of the method at the presence of inference rules and various semantics, Section 7 discusses related work, and finally, Section 8 concludes the paper and identifies issues for further research.

Software and datasets are available to download and use from `http://www.ics.forth.gr/isl/BNodeDelta`.

2 RDF KBs with Blank Nodes

Consider there is an infinite set U (RDF URI references), an infinite set B (blank nodes) and an infinite set L (literals). A triple $(s, p, o) \in (U \cup B) \times U \times (U \cup B \cup L)$ is called an *RDF triple* (s is called the *subject*, p the *predicate* and o the *object*). An RDF Knowledge Base (KB) K, or equivalently an *RDF graph* G, is a set of RDF triples.

For an RDF Graph G_1 we shall use U_1, B_1, L_1 to denote the URIs, bnodes and literals that appear in the triples of G_1 respectively. The *nodes* of G_1 are the values that appear as subjects or objects in the triples of G_1.

The equivalence of *RDF graphs* that contain *blank nodes* is defined in [9] as:

Def. 1 (Equivalence of RDF Graphs that contain Bnodes)
Two RDF graphs G_1 and G_2 are *equivalent* if there is a bijection[1] M between the sets of nodes of the two graphs (N_1 and N_2), such that:

- $M(uri) = uri$ for each $uri \in U_1 \cap N_1$
- $M(lit) = lit$ for each $lit \in L_1$
- M maps bnodes to bnodes (i.e. for each $b \in B_1$ it holds $M(b) \in B_2$)
- The triple (s, p, o) is in G_1 if and only if the triple $(M(s), p, M(o))$ is in G_2. $\diamond$

It follows that if two graphs are equivalent then it certainly holds $U_1 = U_2$, $L_1 = L_2$ and $|B_1| = |B_2|$.

Let us now relate the problem of equivalence with *edit distances*.

Def. 2 (Edit Distance over Nodes given a Bijection)
Let o_1 and o_2 be two nodes of G_1 and G_2, and suppose a bijection between the nodes of these graphs, i.e. a function $h : N_1 \to N_2$ (obviously $|N_1| = |N_2|$). We define the edit distance between o_1 and o_2 over h, denoted by $dist_h(o_1, o_2)$, as the number of additions or deletions of triples which are required for making the "direct neighborhoods" of o_1 and o_2 the same (considering h-mapped nodes the same). Formally, $dist_h(o_1, o_2) =$

$$|\{(o_1, p, a) \in G_1 \mid (o_2, p, h(a)) \notin G_2\}| + |\{(a, p, o_1) \in G_1 \mid (h(a), p, o_2) \notin G_2\}| +$$
$$|\{(o_2, p, a) \in G_2 \mid (o_1, p, h^{-1}(a)) \notin G_1\}| + |\{(a, p, o_2) \in G_2 \mid (h^{-1}(a), p, o_1) \notin G_1\}| \quad \diamond$$

Now recall that if G_1 is equivalent to G_2 then there exists a bijection h such that $(a, p, b) \in G_1 \Leftrightarrow (h(a), p, h(b)) \in G_2$. We will denote this by $G_1 \equiv_h G_2$. It follows that:

Theorem 1 (RDF Graph Equivalence and Edit Distance)
$G_1 \equiv_h G_2 \Leftrightarrow dist_h(o, h(o)) = 0$ for each $o \in N_1$.

[1] A function that is both one-to-one (injective) and onto (surjective).

Obviously the above theorem is useful for the case where the bijection h respects the constraints of Def. 1 (i.e. maps named elements to named elements, and anonymous elements to anonymous).

3 On Finding the Optimal Bnode Mapping

Let us now focus on the case where two KBs, K_1 and K_2, are *not necessarily equivalent* and do contain bnodes. We would like to find a mapping over their bnodes that reduces the size (i.e. the number of change operations) of their delta and allows detecting whether K_1 is equivalent to K_2. Furthermore we want an efficient (tractable at least) method for finding such a mapping.

3.1 RDF/S Differential Functions

[21,22] described and analyzed various differential functions for comparing RDF/S knowledge bases. Each differential function returns a set of primitive change operations, i.e. $Add(t)$ and $Del(t)$ where t is an RDF triple. For the needs of this paper, it is enough to use the differential function Δ_e which is defined as follows ("$-$" denotes set difference): $\Delta_e(K_1 \rightarrow K_2) = \{Add(t) \mid t \in K_2 - K_1\} \cup \{Del(t) \mid t \in K_1 - K_2\}$. We call its output *delta*.

3.2 Bnode Name Tuning and Delta Reduction Size

The basic idea for reducing the delta is the following: if we match a bnode b_1 (of B_1) to a bnode b_2 (of B_2), through a bijection M, then these bnodes can be considered as equal at the computation of delta. For example, if K_1 contains a triple $(b_1, name, Joe)$ and K_2 contains a triple $(b_2, name, Joe)$ and we match b_1 to b_2, then these two triples will be considered equal and thus no difference will be reported. However we should note that in the context of versioning or synchronization services the change operations derived by a differential function *should not* be used as they are. For example, consider $K_1 = \{(b_1, name, Joe)\}$ and $K_2 = \{(b_2, name, Joe), (b_2, lives, UK)\}$ and suppose that we match again b_1 to b_2. In this case a mapping-aware comparison function will return the delta $\{Add((b_2, lives, UK))\}$. If we want to apply it on K_1 then we have to replace b_2 by b_1, i.e we should apply on K_1 the operation $Add((b_1, lives, UK))$, and in this way, we will obtain $K_1' = \{(b_1, name, Joe), (b_1, lives, UK)\}$ which is equivalent to K_2. We call this step *Bnode Name Tuning*, and it actually replaces (renames) in the delta the local names of the bnodes of B_2 by the local names of the matched bnodes in B_1. In this way the delta does not need any *rename* operation (i.e. $rename(b_1, b_2)$) and hence not any particular execution order.

Delta Reduction Size. Bnode matching cannot increase delta size. Without bnode matching any pair of bnodes from different KBs is considered different, and thus all triples to which they participate will be different and reported as change operations in the delta. On the other hand, if two bnodes are matched then the delta size is reduced if they participate to triples with the same predicate and the same other node (i.e. the same *subject* or *object*). In the case where all predicates/nodes of these triples are different, the delta size that will be reported is what will be reported without bnode matching.

3.3 Bnode Matching as an Optimization Problem

Here we formulate the problem of finding a mapping between the bnodes of two KBs as an optimization problem. Let $n_1 = |B_1|$, $n_2 = |B_2|$ and $n = min(n_1, n_2)$. We have to match n elements of B_1 with n elements of B_2, i.e. our objective is to find the unknown part of the bijection M. To be more precise, M a priori contains the mappings of all the URIs and literals of the KBs (URIs and literals are mapped as an identity function as in Def. 1), and its unknown part concerns B_1 and B_2. Suppose that $n = n_1 < n_2$. Let $\mathcal{J}$ denote the set of all possible bijections between B_1 and a subset of B_2 that comprises n elements. The number of all possible bijections (i.e. $|\mathcal{J}|$) is $n_2 * (n_2 - 1) * ... * (n_2 - n_1 + 1)$, i.e. the first element of B_1 can be matched with n_2 elements of B_2, the second with $n_2 - 1$ elements, and so on. Consequently, the set of candidate solutions is exponential in size. Since our objective is to find a bijection $M \in \mathcal{J}$ that reduces the delta size (as regards the "unamed" parts of the KBs), we define the cost of a bijection M as follows:

$$Cost(M) = \sum_{b_1 \in B_1} dist_M(b_1, M(b_1)) \tag{1}$$

Def. 3 (The bijection yielding the less delta size) The best solution (or solutions) is defined as the bijection with the minimum cost, i.e. we define:

$$M_{sol} = \arg_M \min_{M \in \mathcal{J}} (Cost(M)) \qquad \diamond$$

The notation arg_M returns the M in $\mathcal{J}$ that gives the minimum cost.

Theorem 2 (Equivalence and Mapping Cost)
If $G_1 \equiv_{M_{sol}} G_2$ (according to Def. 1) then $Cost(M_{sol}) = 0$.

The proof follows easily from the definitions. It is also clear that the inverse of Th. 2 does not hold (i.e. $Cost(M_{sol}) = 0 \not\Rightarrow G_1 \equiv_{M_{sol}} G_2$) because the cost is based on the distance between the direct neighborhoods of the blank nodes *only*, and not between the named parts of the graphs.

From the algorithmic perspective, one naive approach for finding the best solution (i.e. M_{sol}) would be to examine the set of all possible bijections. That would require at least $n!$ examinations (true if $n_1 = n_2 = n$, while if $n_1 < n_2$ then their number is higher than $n!$). However, the problem is intractable in general:

Theorem 3. Finding the optimal bijection (according to Def. 3) is NP-Hard.

Proof*:*
We will show that subgraph-isomorphism *(which is NP-complete problem) can be reduced to the problem of finding the optimal bijection (meaning that our problem is at least as hard as* subgraph-isomorphism*). Let us make the* hypothesis *that we can find the optimal bijection in polynomial time. We will prove that if that hypothesis were true, then we would be able to solve the subgraph isomorphism in polynomial time. The* subgraph isomorphism decision problem *is stated as: given two plain graphs G_1 and G_2 decide whether G_1 is isomorphic to a subgraph of G_2. Let $G_1 = (N_1, R_1)$ and $G_2 = (N_2, R_2)$. We can consider these graphs as two RDF graphs such that: all of their nodes are bnodes and all property edges have the same label. Assume that $|N_1| \leq |N_2|$ and let $n = \min(|N_1|, |N_2|)$. If we can find in polynomial time whether there is a bijection between the n nodes of G_1 and n nodes of G_2 such that $Cost(M_{sol}) = 0$, then this means that we have found whether G_1 is isomorphic to a subgraph of G_2. Specifically, to decide whether there is a subgraph isomorphism, (a) we compute the optimal bijection, say M_{sol}, and (b) we compute its cost. If the cost returned by step (b) is 0 then we return YES, i.e. that there is a subgraph isomorphism. Otherwise we return NO (i.e. there is no subgraph isomorphism). Note that step (a) is polynomial by* hypothesis*, while step (b) relies on Def. 2 and its cost is again polynomial. Regarding the latter, note that M_{sol} contains n pairs, and to compute $dist_M(b_1, b_2)$ for each (b_1, b_2) pair of M, we consider only the direct neighborhoods of the two nodes in the two graphs (for G_2 we have to consider only those that connect nodes that participate in M_{sol})[2]. It follows that its computational cost is analogous to the number of edges of the graphs, and thus polynomial. Therefore given a bijection M_{sol}, to compute $Cost(M_{sol})$ requires polynomial time. Also note that Th. 1 holds also for plain graphs assuming a distance function over not labeled edges. We conclude that if our hypothesis were true, then we would be able to decide subgraph isomorphism in polynomial time.*

We conclude that finding the optimal bijection is NP-Hard. ◇

Below we will show that there are algorithms of polynomial complexity for a frequently occurring case. For the general case, we will propose algorithms of polynomial complexity that return an approximate solution.

3.4 Polynomially-Solved (and Frequently Occurring) Cases

Consider the KBs in Figure 2 and suppose that we want to compute $dist_h(\_:1, \_:6)$ (according to Def. 2). It is not hard to see that this distance depends on the mappings (by h) of the bnodes that are connected to $\_:1$ and $\_:6$, i.e. on the mappings of $\_:3, \_:4, \_:8$ and $\_:9$. However several datasets do not have directly connected bnodes. For this reason, here we study a variation of the problem that is appropriate for this case. The key point is that the distance between two bnodes does not depend on how the rest bnodes are mapped.

This is very important because in this case we can solve the optimization problem (as defined in Definition 3) using the *Hungarian algorithm* [12] (for short Alg_{Hung}, an algorithm for solving the *assignment problem*. Here the elements (bnodes) of B_1 play the role of *workers*, the elements (bnodes) of B_2 play

[2] Alternatively, if $Cost(M_{sol}) \neq 0$ (using the distance as defined in the main paper), we return YES only if $\Delta_e(G_1 \rightarrow G_2)$ as defined in section 3.1, after bnode name tuning, contains only triples each containing one bnode in B_1 and one not in B_1.

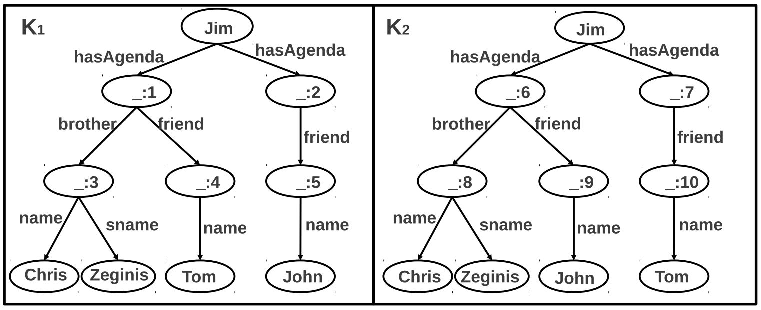


Fig. 2. Two KBs with directly connected bnodes

the role of *jobs*, and the edit distances of the pairs in $B_1 \times B_2$ play the role of the *costs*. Consider for the moment that $|B_1| = |B_2|$. If we compute the edit distances between all possible n^2 pairs, then Alg_{Hung} can find the optimal assignment at the cost of $O(n^3)$ time. This means that finding the optimal solution costs polynomial time. An extension of Alg_{Hung} giving the ability to assign the problem in rectangular matrices (i.e. when $|B1| \neq |B2|$) is already provided in [2]. We conclude that if there are not directly connected bnodes then the optimal mapping can be found in polynomial time.

Theorem 4. Finding the optimal bijection (according to Def. 3) is a polynomial task if there are no directly connected bnodes. $\diamond$

4 Bnode Matching Algorithms

At section 4.1 we present a variation of Alg_{Hung} for getting an *approximate* solution for the general case, then at Section 4.2 we present a *signature*-based algorithm appropriate for larger datasets.

4.1 Hungarian BNode Matching Algorithm

We have already stated that Alg_{Hung} can find the optimal mapping in polynomial time if no directly connected bnodes exist in the compared KBs. For the cases where there are directly connected bnodes, Alg_{Hung} enriched with an assumption regarding how to treat the connected bnodes at the computation of $dist_h$, could be used for producing an *approximate* solution. Also in this case the algorithm will make $n_1 \times n_2$ distance computations (where $n_1 = |B_1|$ and $n_2 = |B_2|$), and the complexity of the algorithm will be again $O(n^3)$.

Regarding connected bnodes, at the computation of $dist_h$, one could either assume that all of the connected bnodes are different, or all of them are the same. The first assumption does not require any bijection (h contains only the identity functions of the URIs and literals). According to Definition 2, the fact that all the bnodes are different means by extension that the triples in the direct neighborhoods connecting blank nodes are different too, even in the case where these triples have the same properties. For instance, applying the Definition 2 between bnodes ($\_:1$, $\_:6$) and ($\_:1$, $\_:7$) of Figure 2, we get that $dist_h(\_:1, \_:6) = 4$ and $dist_h(\_:1, \_:7) = 3$ respectively. However, bnodes $\_:1$, $\_:6$ have two outgoing triples with exactly the same properties, while bnodes $\_:1$, $\_:7$ have only one. We observe that this assumption is not very good because we would prefer $\_:1$ to be "closer" to $\_:6$ than to $\_:7$.

According to the alternative assumption, when comparing bnodes ($\_:1, \_:6$) in Figure 2, bnode $\_:3$ can be matched either with bnode $\_:8$ or with bnode $\_:9$, depending on the existence of a common property between them. This yields $dist_h(\_:1, \_:6) = 0$ since both bnodes have two outgoing triples with common properties (i.e. $(\_:1, brother, \_:3)$ is matched with $(\_:6, brother, \_:8)$ and $(\_:1, friend, \_:4)$ is matched with $(\_:6, friend, \_:9)$). Regarding $\_:1$ and $\_:7$, we get $dist_h(\_:1, \_:7) = 1$ because of the deleted triple $(\_:1, brother, \_:3)$. It follows that the results of this assumption are better over this example, as $\_:1$ is "closer" to $\_:6$ than to $\_:7$. In general it is better because it exploits common properties, and therefore we adopt this assumption in our experiments.

4.2 A Fast ($O(n \log N)$) Signature-Based Algorithm

The objective here is to devise a faster mapping algorithm that could be applied to large KBs, at the cost of probably bigger deltas. We propose a *signature-based* mapping algorithm, for short Alg_{Sign}, which consists of two phases: the signature construction and the mapping construction phase. For each bnode b we produce a string based on the direct neighborhood of b. This string is called the *signature* of bnode b. This phase gives us two lists of signatures, one for the bnodes of each KB. These lists should be considered as bags rather than sets, as there is a probability that two or more bnodes get the same signature. The probability depends on the way the signature is built (we discuss this later).

The mapping phase takes these two bags of strings and compares the elements of the first bag with those of the second. To make *binary search* possible, both

```
Alg. SignatureMapping
Input: two sets of bnodes $B_1$ and $B_2$,
    where $|B_1| < |B_2|$
Out: a bij. $M$ between $B_1$ and $B_2$
(1)  $M = \emptyset$
(2)  $BS_1 = BS_2 = emptybag$
(3)  for each $b_1 \in B_1$
(4)      $BS_1 = BS_1 \cup \{Signature(b_1)\}$
(5)  for each $b_2 \in B_2$
(6)      $BS_2 = BS_2 \cup \{Signature(b_2)\}$
(7)  $sort(BS_1)$
(8)  $sort(BS_2)$
(9)  for each $bs_1 \in BS_1$
(10)     $bs_2 = Lookup(BS_2, bs_1)$
(11)     if ($bs_2 == bs_1$) // exact match
(12)         $M = M \cup \{(bn_1[bs_1], bn_2[bs_2])\}$
// $bn_1[str]$ returns the $b \in B_1$ corresponding to $str$
(13)         $BS_2.remove(bs_2)$
(14)         $BS_1.remove(bs_1)$
(15) for each $bs_1 \in BS_1$
(16)     $bs_2 = Lookup(BS_2, bs_1)$ // closest match
(17)     $M = M \cup \{(bn_1[bs_1]), bn_2[bs_2])\}$
(18)     $BS_2.remove(bs_2)$
(19) return $M$
```

Fig. 3. Alg. The Signature-based bnode matching algorithm

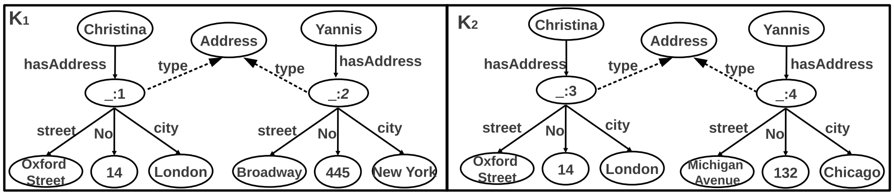


Fig. 4. Two versions of an address Knowledge Base

bags are sorted lexicographically. In particular, we start from the smaller list, say BS_1, and for each string bs_1 in that list we perform a lookup in the second list BS_2 using *binary search*. If an *exact match* exists (i.e. we found the string bs_1 also in BS_2) we produce a mapping, i.e. the pair $(bn_1[bs_1], bn_2[bs_1])$. Since more than one bnodes may have the same signature we select one. We prefer the order as provided by the managing software, which in many cases reflects the order by which bnodes appear in files. As there is a high probability for subsequent versions to keep the same serialization, using the original order increases the probability of matches in case of same signatures[3]. We continue in this way for all strings of BS_1. For each element bs_1 of BS_1 for which *no exact match* was found in BS_2 we perform a second lookup over the remainder part of BS_2, say BS_2', which will produce a mapping based on the *closest* element of BS_2' to the bs_1 element. Specifically, we will match bs_1 to the element of BS_2' to which binary search stopped, i.e. to the *lexicographically closer* element. Note that we perform the closest matches after finishing with the exact matches in order to avoid the situation where an approximate match deters an exact match at a later step.

The complexity of this algorithm is $O(n \log N)$ where $N = \max(n_1, n_2)$ and $n = \min(n_1, n_2)$, assuming that the average graph degree of bnodes (and thus signature size) does not depend on N. The algorithm is shown in Figure 3 and relies on an algorithm *Signature* for producing signatures, and on a algorithm *Lookup* as described earlier. As regards the signature construction method we would like to derive strings that allow matches that will yield small deltas even if the bnodes do not match exactly. To this end, we should give priority (i.e. bring to the front part of the string) to the items of the bnode that have lower probability to be changed from one version to the other.

Table 1. *Signatures* on bnodes of K_1 and K_2 of Fig. 4 according to the given option

Local Name	Signature
_ : 1	$ChristinahasAddress\diamond typeAddress\diamond cityLondon * No14 * streetOxfordStreet$
_ : 3	$ChristinahasAddress\diamond typeAddress\diamond cityLondon * No14 * streetOxfordStreet$
_ : 2	$YannishasAddress\diamond typeAddress\diamond cityNewYork * No445 * streetBroadway$
_ : 4	$YannishasAddress\diamond typeAddress\diamond cityChicago * No132 * streetMichiganAvenue$

[3] We do the same in Alg_{Hung} in case of ties in costs.

Consider bnode $\_:1$ of Figure 4 which is involved in the following triples: *Incoming*: $\{(Christina, hasAddress, \_:1)\}$, *Outgoing*: $\{(\_:1, street, OxfordStreet)$, $(\_:1, No, 14)$, $(\_:1, city, London)\}$, *Class Type*: $\{(\_:1, type, Address)\}$. Each of these triples will be mapped to a substring (e.g. "ChristinahasAddress" for the triple $(Christina, hasAddress, \_:1)$). The set *Class Type* contains the triples with the `rdf:type` ("type" in the figure) property of the respective bnode. For the three different sets of triples (Incoming, Outgoing, Class Type) we are going to construct three sets of substrings respectively. The substrings inside each set are sorted lexicographically and separated by a special character, here denoted by $*$. The concatenation of these sets of substrings will yield the signature. A key point is the order by which the sets are concatenated. One option is to give a first priority to the set of the incoming triples, a second priority to the set with type information (i.e. "typeAddress"), and the last priority to the set of the outgoing triples. We should also mention that inside the signature the sets are separated by a special character, here denoted by $\diamondsuit$. Table 1 shows the signatures of all the bnodes of Figure 4 according to this option. The proposed ordering of the substrings inside the signature stems from the assumption that the probability for the outgoing statements to change is higher than the incoming (e.g. in Figure 4 updating the address of a person is more probable than changing his/her name). Under this assumption the incoming statements should precede the outgoing inside the signature. Similarly for the class type of the bnode, it is not usual to be changed from one version to the other.

We represent the blank nodes which are subjects of incoming statements or objects of the outgoing statements, by a special character $\clubsuit$ (i.e. we treat them as equal, as we did in the 2nd assumption of approximation version of Alg_{Hung}).

5 Experimental Evaluation

Real Datasets. We performed experiments for evaluating the potential for delta reduction, equivalence detection and time efficiency. In our experiments[4], we used two real datasets available in the LOD cloud: the *Swedish open cultural heritage* dataset[5], and the *Italian Museums* dataset[6], published from LKDI[7]. From each one we downloaded two versions with a time difference of one week or month. A preprocessing was necessary for corrections (e.g. missing URIs for some classes) and for merging the files. The features of these two datasets are given in Table 2. In both datasets there are *no directly connected bnodes*.

Experiments were conducted *with* and *without* bnode mapping. Regarding mapping we tested: (a) the *random*, (b) the *Hungarian*, and (c) the *Signature*-based mapping methods. The results are shown in Table 3. The first rows show

[4] Using Sesame RDF/S Repository (main memory), using a PC with Intel Core i3 at 2.2 Ghz, 3.8 GB Ram, running Ubuntu 11.10.

[5] `http://thedatahub.org/dataset/swedish-open-cultural-heritage` used from `http://kringla.nu/kringla/` for providing information on cultural data of Sweden.

[6] `http://thedatahub.org/dataset/museums-in-italy`

[7] `http://www.linkedopendata.it/`

Table 2. Features of two real LOD datasets

	Swedish		Italian	
	File 1	File 2	File 1	File 2
Date	15/10/11	22/10/11	2/11/11	4/12/11
$\|Triples\|$	3,750	3,589	49,897	49,897
$\|BNodes\|$	535	509	6,390	6,390
$\|Triples$ with $bnodes\|$	77.7%	77.2%	43.85%	43.85%
Total Size	378 KB	365 KB	5.49 MB	5.46 MB

Table 3. Experimental results over real datasets

	Swedish				Italian			
	without BM	*with BM (bnode matching)*			*without BM*	*with BM (bnode matching)*		
		Random	Alg_{Hung}	Alg_{Sign}		*Random*	Alg_{Hung}	Alg_{Sign}
$\|Added\|$	2,805	2,726	75	127	21,885	19,762	3	3
$\|Deleted\|$	2,966	2,887	236	288	21,885	19,762	3	3
$\|\Delta_e\|$	5,771	5,613	311	419	43,770	39,524	6	6
BLoad Time(ms)	-	631	630	634	-	428	423	421
SC Time(ms)	-	-	-	210	-	-	-	840
BM Time(ms)	-	1.3	5,391	130	-	4.9	576,592	82.5
Diff Time(ms)	55	64	30	47	145	166	169	163
Tuning Time(ms)	-	15	0.2	0.5	-	3,332	9.4	9.5
Total Time(ms)	57	715	5,931	1,024	147	3,935	577,197	1,521

the size of the yielded deltas and the last rows the time required for loading the bnodes (BLoad), constructing signatures (SC), bnode maping (BM), delta computation (Diff), bnode name tuning (Tuning Time), and the total time. We observe that the algorithms provide a delta of 12.7 to $7,294$ times smaller than without bnode mapping. Alg_{Hung} yields an equal (for the Italian) or smaller (0.34 times smaller for the Swedish) delta than Alg_{Sign}, but it requires more time (from 15 to 624 times).

Synthetic Datasets. Although semantic data generators already exist in the bibliography, none of them deals with the blank node connectivity issues. Therefore we designed and developed a synthetic generator over the UBA (Univ-Bench Artificial data generator) [5] that can generate datasets with the desired bnode structures. Each dataset corresponds to an RDF graph G. Let $Nodes$ be the set of all nodes in the graph, B be the set of bnodes ($B \subseteq Nodes$), and $conn(o)$ be the nodes of G that are directly connected with a node $o \in Nodes$. We define $b_{density}$ as $b_{density} = \text{avg}_{b \in B} \frac{|conn(b) \cap B|}{|conn(b)|}$. Note that if there are no directly connected bnodes then $b_{density} = 0$. The extended generator can create datasets with the desired $b_{density}$ and the desired maximum length of paths that consist of edges that connect bnodes (we denote by b_{len} their average). Using the synthetic generator, we created a sequence of 9 pairs of KBs (each pair has two subsequent versions of a KB). For instance, the first KB is K_0 and its pair is K'_0. Each time we compare the subsequent versions of a pair with respect to mapping time and yielded delta size. From now on we express the delta size as a percentage of the number of triples of the KB, i.e. as $\frac{|\Delta_e(K,K')|}{\frac{|K|+|K'|}{2}}$. Table 4 shows the blank node properties of each pair of KBs, its optimal delta size over its

Table 4. Blank node Features of the synthetic dataset

K	$\|triples\|$	$\|B\|$	D_a	$b_{density}$	b_{len}	Optimal delta size	Variation
K_{0a}	5,846	240	13.4	0	0	1%	No connected blank nodes
K_{1a}	5,025	240	10.5	0.1	1	0.5%	*b_Neighborhoods* of 2 bnodes, reduced *b_named* triples
K_{2a}	2,381	240	7	0.15	1	1.5%	Reduced *b_named* triples
K_{3a}	1,628	240	5	0.2	1	1.5%	Reduced *b_named* triples
K_{4a}	1,636	240	5	0.2	1.15	1%	*b_Neighborhoods* of up to 8 bnodes
K_{5a}	1,399	240	4	0.25	1.15	1.7%	Reduced *b_named* triples
K_{6a}	919	240	3	0.32	1.15	3.2%	*b_Neighborhoods* of up to 15 bnodes, reduced *b_named* triples
K_{7a}	909	240	3.25	0.4	1.35	2.7%	Connect *b_Neighborhoods*, reduced *b_named* triples
K_{8a}	1,001	240	3.94	0.5	21.5	2.5%	Connect *b_Neighborhoods*

subsequent version (known by construction) and its variation over the next pair of KBs (we call *b_Neighborhood* every subgraph having as nodes only bnodes, and we call *b_named* triple every triple that contains one bnode). With D_a we denote the average number of direct edges of the bnodes (i.e. average number of triples to which a bnode participates).

Figure 5(left) gives the delta reduction potential of each algorithm in logarithmic scale. Without bnode mapping the delta size ranges from 95% (for the second pair of KBs) to 143% (for the ninth pair of KBs). Instead for Alg_{Hung} it ranges from 0.47% to 10.67% and for Alg_{Sign} it ranges from 1% to 11.5%. Notice that Alg_{Sign} does not reduce the delta to the optimal for any pair of datasets, while Alg_{Hung} achieves the optimal delta for most of the pairs.

Figure 5 (right) shows the delta reduction potential for the same pairs with the difference that the two bnode lists are not scanned in the original order (as in the left figure), but the second list is reversed. We notice that as the areas of directly connected bnodes become bigger (after the sixth pair of datasets), we get different (here higher) deltas. In such areas the direct neighborhoods lose their discrimination ability and thus the delta reduction potential becomes more unstable, increasing the probability to get a bigger delta.

If we use the optimal delta as baseline, and compute the percentage $\frac{|\Delta_x|-|\Delta_{opt}|}{|\Delta_{opt}|}$, in the first diagram this percentage for Alg_{Hung} falls in [0, 2.88], while the Alg_{Sign}'s percentage falls in [0.4,3.2] (in the second diagram they fall in [0,8] and [0.4,8] resp.).

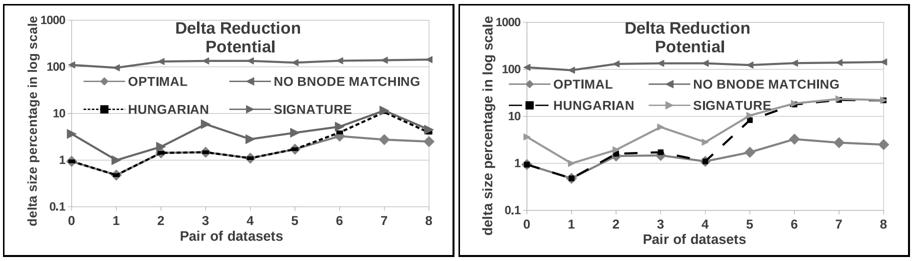


Fig. 5. Delta Reduction over the synthetic datasets

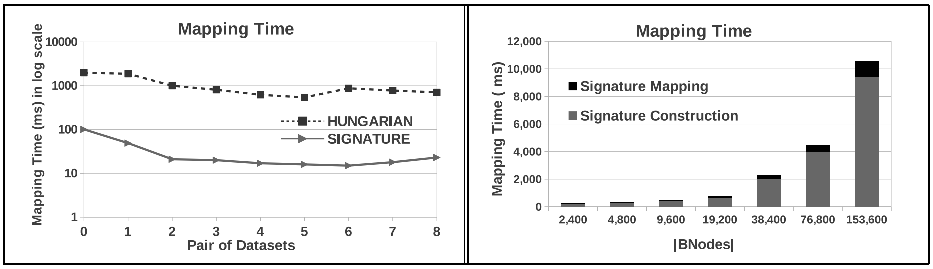


Fig. 6. Mapping times over the synthetic datasets

Figure 6 (left) shows the mapping times of each algorithm in logarithmic scale. Alg_{Sign} gives two orders of magnitude lower mapping times.

Equivalences. Regarding equivalent KBs, if there are no directly connected bnodes then Alg_{Hung} detects them at polynomial time (recall Th. 4). To investigate what happens if there are directly connected bnodes we compared the pairs (Kia, Kia) for i=0 to 8 of the synthetic KBs. In case of similarly ordered bnode lists both Alg_{Hung} and Alg_{Sign} detected equivalences for all the KBs, while for reverse scanned bnodes lists they detected 5 of the 9 equivalences. They did not detect equivalences for the KBs with $b_{density} \geq 0.25$.

Bigger Datasets. To investigate the efficiency of Alg_{Sign} in bigger datasets, we created 7 pairs of KBs: the first pair contains 23,827 triples and 2,400 bnodes, the second pair has the double number of triples and bnodes, and so on, until reaching the last pair containing 153,600 bnodes. From Fig. 6 (right) we can see that the mapping time for Alg_{Sign} was only 10.5 seconds for the seventh pair of KBs (153,600 bnodes). Alg_{Hung} could not be applied even to the third pair of KBs due to its high (quadratic) requirements in main memory space.

The results are summarized in the concluding section.

Measuring the Approximation. The upper bound of the reduction of the delta size that can be achieved with bnode matching is the min number of bnodes of the two KBs multiplied by their average degree. Experimentally we have investigated whether $b_{density}$ (which is zero if there are no directly connected bnodes, and equal to 1 if all nodes are bnodes as in the proof of Th. 3), is related with the deviation from the optimal delta $d_x = \frac{|\Delta_x| - |\Delta_{opt}|}{|\Delta_{opt}|+1}$. Results over equivalent and non-equivalent KBs are shown at Figure 7. Both algorithms give a much smaller deviation from optimal than without bnode matching (its d_x ranges [47,114]). We also observe that keeping the original order of the bnodes is beneficial for both algorithms. For the non equivalent KBs the Alg_{Hung} gives always equal or (mostly) smaller delta than the Alg_{Sign}, while for the equivalent both algorithms give exactly the same deviation.

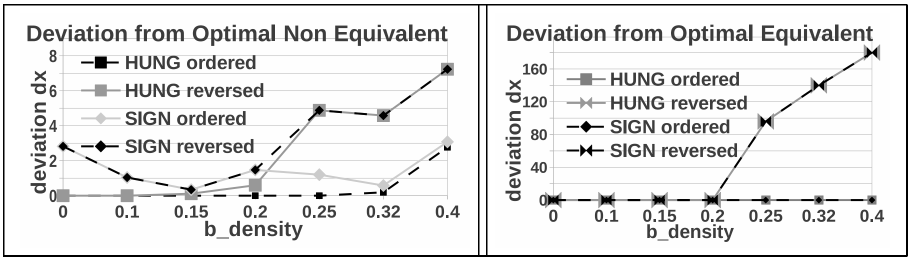


Fig. 7. d_x over non equivalent (left) and equivalent (right) KBs

6 Discussing Semantics and Inference Rules

Apart from the explicitly specified triples of a KB, other triples can be inferred based on the RDF/S semantics [6], or other custom inference rules. In some cases one may want to decide whether two KBs are equivalent or to compute their delta with respect to a particular set of rules. In such scenarios, equivalence can be based again on the Def. 1 and the edit distance over nodes on the Def 2 with the only difference that the graphs should be *completed* according to the inferred triples. It follows that if the semantics is based on a set of inference rules yielding a finite closure, then the graph is finite and thus our method can be applied. Some semantics offering finite closures are RDF/S semantics, Minimal RDFS semantics [11], ter Horst's pD* semantics and OWL 2 RL, or even application-specific like [18].

It is worth mentioning, that the optimal bnode mapping over the *complete* graphs may be different from the optimal mapping when considering the *explicit* graphs. In the example of Figure 8, where fat arrows denote `rdfs:subClassOf` relationships and dotted arrows `rdf:type` relationships, the bijection with the minimum cost over the explicit graphs (left) is {(_:1,_:4),(_:2,_:3)}, while at the completed graphs (right) the bijection with the minimum cost is {(_:1,_:3),(_:2,_:4)}

Furthermore, for checking equivalence (at the presence of bnodes) or computing deltas, one could use the *reduced graphs* in case they are unique (note that the *reduction* of a K_a, is the smallest in size K_b that is equivalent to K_a, i.e. K_a and K_b have the same closure).

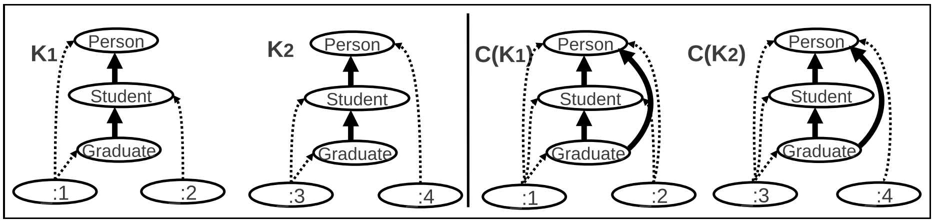


Fig. 8. Comparing the explicit versus the complete graphs of two KBs

7 Related Work

Jena [3] provides a method for deciding whether two KBs that contain bnodes are equivalent (assuming Def. 1) and the adopted algorithm is GI-Complete. PromptDiff [16] and Ontoview [8] employ heuristic matchers to decide whether two bnodes from different KBs match or not, while CWM [1] is able to match two blank nodes only if they have functional term labels. Semversion [20] creates and assigns unique identifiers to bnodes so that to be able to identify the matching bnodes across versions. However, this is possible only if all versions have been derived from the same system. RDFSync [19] aims at fast synchronization, i.e. at reducing the parts of the KBs that have to be compared, and no effort is dedicated for finding a bnode mapping for reducing the delta size. [13] introduced a blank node mapping with $O(n^2)$ complexity aiming at merging sets of RDF triples (RDF molecules). However, this mapping presupposes that bnodes are parts of uniquely identified triples. This mapping method is not applicable in the general case and cannot be used for delta reduction. To the best of our knowledge our work is the first one that attempts to find a bnode mapping that reduces the size of deltas between KBs (that are not equivalent). Although there are several works for constructing RDF/S *mappings* (e.g. see [14]), they are not directly related since they map the *named* entities of the two KBs, and thus they take into account lexical similarities, something that is not possible with bnodes.

8 Concluding Remarks

In this paper we showed how we can exploit bnode anonymity to reduce the delta size when comparing RDF/S KBs. We proved that finding the optimal mapping between the bnodes of two KBs, i.e. the one that returns the smallest in size delta regarding the *unnamed* part of these KBs, is NP-Hard in the general case, and polynomial in case there are not directly connected bnodes. To cope with the general case we presented polynomial algorithms returning approximate solutions.

In real datasets with no directly connected bnodes Alg_{Sign} was two orders of magnitude faster than Alg_{Hung} (less than one second for KBs with 6,390 bnodes), but yielded up to 0.34 times (or 34%) bigger deltas than Alg_{Hung}, i.e. than the optimal mapping. Alg_{Hung} also identified all equivalent KBs.
For checking the behavior of the algorithms in KBs with directly connected bnodes, we created synthetic datasets, over which we compared Alg_{Sign} and the Alg_{Hung} *approximation* algorithm. Alg_{Hung} yielded from 0 to 3 times smaller deltas than Alg_{Sign}, but the latter was from 18 to 57 times faster. Alg_{Sign} requires only 10.5 seconds to match 153,600 bnodes.

This is the first work on this topic. Several issues are interesting for further research. For instance, it is worth investigating other special cases where the optimal mapping can be found polynomially (e.g. directly connected bnodes that form graphs of bounded tree width). Another direction is to comparatively evaluate various (probabilistic) signature construction methods and greedy approximation algorithms.

Software and datasets are available to download and use from: `http://www.ics.forth.gr/isl/BNodeDelta`.

Acknowledgement. Many thanks to the anonymous reviewers for their comments which helped us to improve the paper, as well to the members of FORTH-ICS-ISL. This work was partly supported by the NoE *APARSEN* (Alliance Permanent Access to the Records of Science in Europe, FP7, Proj. No 269977, 2011-2014), and the FP7 Research Infrastructures projects *SCIDIP-ES* (SCIence Data Infrastructure for Preservation - Earth Science, 2011, 2014), and *iMarine* (FP7 Research Infrastructures, 2011-2014).

References

1. Berners-Lee, T., Connoly, D.: Delta: An Ontology for the Distribution of Differences Between RDF Graphs (2004), `http://www.w3.org/DesignIssues/Diff`
2. Bourgeois, F., Lassalle, J.-C.: An extension of the Munkres algorithm for the assignment problem to rectangular matrices. Commun. ACM (1971)
3. Carroll, J.J.: Matching RDF Graphs. In: Horrocks, I., Hendler, J. (eds.) ISWC 2002. LNCS, vol. 2342, pp. 5–15. Springer, Heidelberg (2002)
4. Cloran, R., Irwin, B.: Transmitting RDF graph deltas for a Cheaper Semantic Web (2005)
5. Guo, Y., Pan, Z., Heflin, J.: LUBM: A benchmark for OWL knowledge base systems. Selected Papers from the Intern. Semantic Web Conf. ISWC (2004)
6. Hayes, P.: RDF Semantics, W3C Recommendation (2004)
7. Heflin, J., Hendler, J., Luke, S.: Coping with Changing Ontologies in a Distributed Environment. In: AAAI 1999 Workshop on Ontology Management (1999)
8. Klein, M., Fensel, D., Kiryakov, A., Ognyanov, D.: Ontology Versioning and Change Detection on the Web. In: Gómez-Pérez, A., Benjamins, V.R. (eds.) EKAW 2002. LNCS (LNAI), vol. 2473, pp. 197–212. Springer, Heidelberg (2002)
9. Klyne, G., Carroll, J.J.: Resource Description Framework (RDF): Concepts and Abstract Syntax (2004)
10. Mallea, A., Arenas, M., Hogan, A., Polleres, A.: On Blank Nodes. 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. 421–437. Springer, Heidelberg (2011)
11. Muñoz, S., Pérez, J., Gutierrez, C.: Simple and Efficient Minimal RDFS. Web Semantics (2009)
12. Munkres, J.: Algorithms for the assignment and transportation problems. J-SIAM 5(1) (1957)
13. Newman, A., Li, Y.F., Hunter, J.: A scale-out RDF molecule store for improved co-identification, querying and inferencing. In: Intern. Workshop on Scalable Semantic Web Knowledge Base Systems, SSWS (2008)
14. Noessner, J., Niepert, M., Meilicke, C., Stuckenschmidt, H.: Leveraging Terminological Structure for Object Reconciliation. 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. 334–348. Springer, Heidelberg (2010)
15. Noy, N.F., Kunnatur, S., Klein, M., Musen, M.A.: Tracking Changes During Ontology Evolution. In: McIlraith, S.A., Plexousakis, D., van Harmelen, F. (eds.) ISWC 2004. LNCS, vol. 3298, pp. 259–273. Springer, Heidelberg (2004)

16. Noy, N.F., Musen, M.A.: PromptDiff: A Fixed-point Algorithm for Comparing Ontology Versions. In: Procs. of AAAI 2002 (2002)
17. Schandl, B.: Replication and Versioning of Partial RDF Graphs. 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. 31–45. Springer, Heidelberg (2010)
18. Strubulis, C., Tzitzikas, Y., Doerr, M., Flouris, G.: Evolution of Workflow Provenance Information in the Presence of Custom Inference Rules. In: 3rd Intern. Workshop on the Role of Semantic Web in Provenance Management (SWPM 2012), co-located with ESWC 2012, Heraklion, Crete (2012)
19. Tummarello, G., Morbidoni, C., Bachmann-Gmür, R., Erling, O.: RDFSync: Efficient Remote Synchronization of RDF Models. 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.) ISWC/ASWC 2007. LNCS, vol. 4825, pp. 537–551. Springer, Heidelberg (2007)
20. Volkel, M., Winkler, W., Sure, Y., Ryszard Kruk, S., Synak, M.: SemVersion: A Versioning System for RDF and Ontologies. In: Procs. of ESWC 2005 (2005)
21. Zeginis, D., Tzitzikas, Y., Christophides, V.: On the Foundations of Computing Deltas Between RDF Models. 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.) ISWC/ASWC 2007. LNCS, vol. 4825, pp. 637–651. Springer, Heidelberg (2007)
22. Zeginis, D., Tzitzikas, Y., Christophides, V.: On Computing Deltas of RDF/S Knowledge Bases. ACM Transactions on the Web, TWEB (2011)

Hybrid SPARQL Queries: Fresh vs. Fast Results

Jürgen Umbrich, Marcel Karnstedt, Aidan Hogan, and Josiane Xavier Parreira

Digital Enterprise Research Institute, National University of Ireland, Galway
`firstname.lastname@deri.org`

Abstract. For Linked Data query engines, there are inherent trade-offs between centralised approaches that can efficiently answer queries over data cached from parts of the Web, and live decentralised approaches that can provide fresher results over the entire Web at the cost of slower response times. Herein, we propose a *hybrid query execution* approach that returns fresher results from a broader range of sources vs. the centralised scenario, while speeding up results vs. the live scenario. We first compare results from two public SPARQL stores against current versions of the Linked Data sources they cache; results are often missing or out-of-date. We thus propose using *coherence estimates* to split a query into a sub-query for which the cached data have good fresh coverage, and a sub-query that should instead be run live. Finally, we evaluate different hybrid query plans and split positions in a real-world setup. Our results show that hybrid query execution can improve freshness vs. fully cached results while reducing the time taken vs. fully live execution.

1 Introduction

As of today, there are an estimated 30 billion facts published on the Web as Linked Data [3]. Traditional approaches for querying Linked Data rely on optimised, centralised RDF stores that *cache* remote content [2,17,4]. However, maintaining comprehensive and up-to-date cached data is an impossible task. First, the coverage of centralised stores is limited by the amount of data that can be located, retrieved and indexed by local servers. Second, result freshness is determined by the last time the relevant Web documents were cached, which can often be measured in days or even months, leading to stale query answers.

Conversely, various authors have proposed techniques to process queries live and directly over Linked Data [6,20,8,10], bypassing the need for maintaining a replicated (full) store. In live-querying scenarios, coverage spans the Web, and the freshness of results depends on live-query latency, which can generally be measured in terms of minutes or even seconds. However, in live approaches, retrieving remote content from diverse sources at query-time naturally implies much slower response times compared to querying centralised stores.

Thus, as shown in Figure 1, there is an inherent trade-off between query approaches that give fresh results versus approaches that give fast results, represented at two ends by centralised SPARQL stores and live-query techniques respectively. Aside from performance, the *recall* of answers is also crucial to consider. Figure 2 shows how results from a live query approach (L) and results

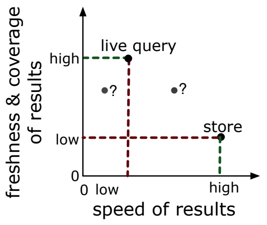


Fig. 1. Query Trade-off

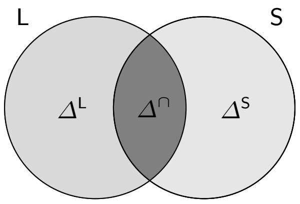


Fig. 2. Result set (Venn) diagram

from a store of cached Web data (S) may diverge. Some of the results remain the same ($\Delta^{\cap}$); when this happens we say that the SPARQL store returned *coherent* results. However, the store may return results that live querying does not (Δ^{S}), which may be stale results from sources that have changed, or may be accurate results from sources that live querying did not consider. Conversely, the live approach may find answers that the store did not (Δ^{L}), either from updates in remote sources, or from sources not cached/indexed by the store.

Evidently, both centralised and live query engines have their inherent strengths and weaknesses. We thus propose a hybrid approach that combines the two. Our hybrid engine can be thought of as a "live-wrapper" for centralised SPARQL endpoints that splits a query into two, where one part is executed over the centralised store and the other part is executed using existing live-querying techniques. By getting the store to quickly service query-patterns for which it has good up-to-date coverage, and by running the rest of the query live, our hybrid approach aims to strike a balance between fresh and fast results.

Deciding which parts of the query to run live and which to run centrally requires knowledge of the *coherence* of the replicated store wrt. remote data, which depends on the dynamicity of remote data and coverage of the store. Consider the hypothetical query WHAT ARE THE CURRENT TEMPERATURES IN EUROPEAN CAPITAL CITIES? Data about current temperatures are dynamic and appropriate for live-query techniques (the store would return stale results). Also, the store may not have indexed data about which continent each city is on; this part should also be fetched live. Conversely, information about capital cities is static and well-covered by the store. In our hybrid approach, up-to-date results about capital cities are quickly retrieved from the store and enriched with results from the Web about continents and temperatures. Stale results are avoided by not asking the centralised store about current temperatures.

We continue this paper with background on Linked Data querying (Section 2). We then introduce our hybrid query architecture (Section 3). Next, we propose probing RDF stores to collect coherence estimates, and present experiments for two public SPARQL engines (Section 4). We then detail our hybrid query-planning component (Section 5). For the two public stores, we evaluate different hybrid query plans and the extent to which they speed up live querying while freshening up centralised results (Section 6). We then conclude (Section 7).

2 Background

Centralised Linked Data stores execute SPARQL queries over a local copy of Web documents. Some endpoints, like FactForge [2], index selected subsets of Linked Data. Other systems, like OpenLink's LOD cache[1] and the Sindice [17] SPARQL endpoint[2] (both powered by Virtuoso [4]) aim to have broad coverage of Linked Data. However, as we show in Section 4, constantly maintaining a broad, fresh coverage of remote data is unfeasible in such setups.

Recently, various authors have proposed methods for performing live querying, accessing remote data *in situ* and at runtime. Ladwig and Tran [8] categorise these approaches as: (i) top-down, (ii) bottom-up, and (iii) mixed strategy. Top-down evaluation determines remote, query-relevant sources using a *source-selection index*: a local repository summarising information about sources that can vary from inverted-index structures [17,10], to query-routing indexes [16], schema-level indexes [15], or hash-based summaries [20]. The bottom-up query evaluation strategy discovers relevant sources on-the-fly during the evaluation of queries by selectively and recursively following links starting from a "seed set" of URIs taken from the query [6]. The third strategy uses (in a top-down fashion) some knowledge about sources to generate the seed list, then discovering additional relevant sources using a bottom-up approach [8]. These three approaches rely on time-consuming remote lookups, but conversely offer fresh results.

In this paper, we use the bottom-up approach proposed by Hartig et al. [6], called link-traversal based query execution (LTBQE), to provide live results. LTBQE uses dereferenceable URIs in a query to find remote documents relevant to that query. During query execution, dereferenceable links that match query patterns are followed to discover further relevant data. We choose LTBQE for live querying as it requires no local knowledge and thus, in the hybrid scenario, can find sources that the store may not even be aware of. An inherent weakness of LTBQE is its dependence on the availability of dereferenceable URIs. However, as the uptake of Linked Data principles continues, we expect the ratio of dereferenceable data on the Web to increase. We refer the reader to our previous work which analyses the prevalence of dereferenceable URIs and the ratio of information about RDF resources that is dereferenceable on the Web [19].

While the above approaches access data directly, an orthogonal approach to live querying is that of federated SPARQL, where queries are executed over a group of possibly remote endpoints [12,13,1]. Given the recent proliferation of SPARQL endpoints on the Web of Data [3], federation is a timely topic that enjoys increasing attention [5]. Our approach could also use live federated techniques, though we would need to ascertain the freshness of remote endpoints.

Like us, various authors have discussed the combination of local (central) and remote (live) querying techniques on a theoretical, optimisation, engineering and social level (e.g., [9,22]). However, to the best of our knowledge, no-one has tackled the question of deciding which parts of a query are suitable for local/remote execution; here, we propose making such a split based on dynamicity estimates.

[1] http://lod.openlinksw.com/sparql

[2] http://sparql.sindice.com/

In this light, our approach relates to the broad field of research on (Web) caching and the problem of guaranteeing cache coherence [11], as well as semantic caching in, e.g., mediator systems [7]. However, instead of splitting queries, such systems often apply an all-or-nothing approach, either relying solely on locally cached data or going entirely live. More recently, various authors have discussed invalidation of SPARQL caches [23] but rely on monitoring inserts within the local system, and are not concerned with the dynamics of remote data.

The work in hand is based on the idea of hybrid SPARQL queries originally proposed by us in [22]. In [21], we discussed a wide range of general strategies to implement the proposed hybrid query planning and processing approach. The current paper is based on the conclusions drawn therein and proposes a first concrete instantiation of an according query engine including a comprehensive experimental evaluation over real datasets.

3 Architecture of a Hybrid Query Engine

Our proposed hybrid query engine has the following targets: $\textcircled{T1}$ **fast response times** close to those of centralised queries; $\textcircled{T2}$ **coherence of results** close to those of live query processing such that we retrieve fresh answers; $\textcircled{T3}$ **system independence**, i.e., being compatible with any SPARQL-enabled store or live-query processor; and $\textcircled{T4}$ **lightweight implementation** with low resource requirements, particularly regarding main memory.

As previously discussed, $\textcircled{T1}$ and $\textcircled{T2}$ are antagonist targets: thus, the main component in the architecture is the query planner which tries to find an overall "optimal" trade-off for a given request, deciding what parts of the query to delegate to the live engine and to the store. With regards to $\textcircled{T3}$, we can initialise our architecture with an index query interface and a live query interface as black-box components; both consume SPARQL queries and produce SPARQL results, but the former interfaces with a central store, whereas the latter interfaces with the Web. Finally, to help find that trade-off, the coherence monitor collects *high-level* empirical statistics (see Section 4) about the store's coverage of data for different query patterns compared with the Web. These compact estimates have (relatively) low maintenance costs (as per $\textcircled{T4}$). The resulting architecture is illustrated in Figure 3.

The index query interface can be a (possibly remote) public SPARQL store or any data warehousing approach that offers the SPARQL protocol (e.g., an intranet database). The live query interface also accepts SPARQL queries and could be based on, for instance, a bottom-up link-traversal engine [6] or a top-down source selection index [20], or some combination thereof. Here we instantiate the live query processor with a bottom-up, link-traversal based query execution approach (LTBQE) as originally proposed by Hartig et al. [6].

The coherence monitor collects information about the coverage and freshness of different triple patterns and sources. The coherence estimates of individual patterns is used by the query planner component to split a given query into two sub-queries—a central and a live sub-query. Eventually, the query processor forwards the central part to the index query interface and the live part

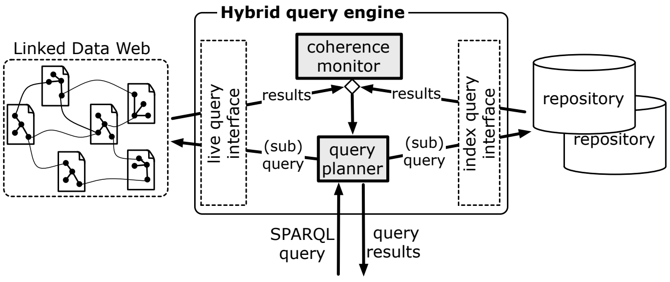


Fig. 3. Architecture of a hybrid query-engine

is processed over the relevant Web sources *in situ*. We see this conceptually straightforward architecture as a first step towards freshening up centralised results: topics such as adaptive coherence estimates and more fine-grained interaction between the central and remote query processors are left to future work.

Note that since the central SPARQL store is treated as a black-box (as per (T3)), we cannot influence the design of the physical plan for the static sub-query: we delegate generating the final sub-query plan to the engine, which we assume implements, e.g., local selectivity estimates to organise optimal execution. In the general case, a similar situation exists for the live query processor.

In the following sections, we elaborate further on the COHERENCE MONITOR component (Section 4) and the QUERY PLANNER component (Section 5).

4 Coherence Estimation

Given the scope [3] and dynamicity [18] of Linked Data, results returned by a centralised endpoint are inherently limited by its coverage of the Web and by the freshness of its local index. The COHERENCE MONITOR computes and stores the coherence estimates of query patterns for a centralised endpoint in contrast to the fresh results given by live query execution.

For a given endpoint, we issue the same set of queries against both the store and the live engine and compare the results, identifying data predicate–source combinations that are likely to be stale. For testing, we chose the two aforementioned SPARQL stores covering a broad range of Linked Data: “the Semantic Web Index” hosted by Sindice, and the “LOD Cache” hosted by OpenLink. We randomly sampled 12,000 URIs from the 2011 Billion Triple Challenge dataset[3], which covers around 8 million RDF Web documents.[4] For each URI, we generated the following query.

[3] `http://challenge.semanticweb.org/`

[4] We considered using the SPARQL 1.1 `SAMPLE` keyword, but Virtuoso does not support SPARQL 1.1 (though it does support similar custom syntax). Further, `SAMPLE` makes no guarantees about the randomness of results.

```
SELECT ?sIn, ?pIn, ?oOut, ?pOut
WHERE { ?sIn ?pIn <entityURI> . <entityURI> ?pOut ?oOut . }
```

This query returns all values of RDF triples in which the given entity URI appears in either the subject or object position. We then compare the set of store results (S) to the set of live results (L). We view results as consisting of sets of sets of variable bindings (i.e., $\mathsf{S}, \mathsf{L}, \Delta^* \subset 2^{\mathbf{V} \times \mathbf{UL}}$ reusing common notation for the set of all query variables, URIs and literals resp.); we exclude answers that involve blank nodes to avoid issues of scoping and inconsistent labelling.

We reuse notation outlined in Figure 2. $\Delta^{\cap} := \mathsf{L} \cap \mathsf{S}$ refers to the set of results in both L and S, i.e., results for which the store is up-to-date. $\Delta^{\mathsf{L}} := \mathsf{L} \setminus \mathsf{S}$ refers to the set of results returned by the Web that are not returned by the store. $\Delta^{\mathsf{S}} := \mathsf{S} \setminus \mathsf{L}$ indicates results returned only by the store. We add subscripts to indicate results for a certain query, e.g., Δ^{L}_q. We denote results for a predicate p as, e.g., $\Delta^{\mathsf{L}}_q(p) := \{r \in \Delta^{\mathsf{L}}_q : (\texttt{?pIn}, p) \in r \vee (\texttt{?pOut}, p) \in r\}$, and say $p \in \Delta^{\mathsf{L}}_q$ iff $\Delta^{\mathsf{L}}_q(p) \neq \emptyset$.

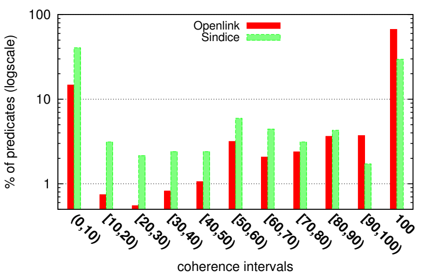
To ensure lightweight statistics with broad applicability, our notion of coherence for query patterns centres around predicates. This restricts our approach to triple patterns with a constant in the predicate position; other patterns are assigned a default estimate. We ran our experiments in early March 2012 and gathered coherence information for 2,550 predicates for OpenLink and 1,627 predicates for Sindice.[5] To quantify the coherence of individual predicates based on the results, we define the *result-based coherence* measure, which computes the ratio of missing results for a predicate p. For the full set of queries $\mathcal{Q}$, let $M_r(p)$ denote the count of all live results involving the predicate p that were missed by the store, summated across all queries ($M_r(p) = \sum_{q \in \mathcal{Q}} |\Delta^{\mathsf{L}}_q(p)|$). Let $L_r(p)$ denote the count of all results involving p retrieved from the live engine ($L_r(p) = \sum_{q \in \mathcal{Q}} |\mathsf{L}_q(p)|$). Result-based coherence is then:

$$\mathsf{coh}_r(p) = 1 - \frac{M_r(p)}{L_r(p)} \quad .$$

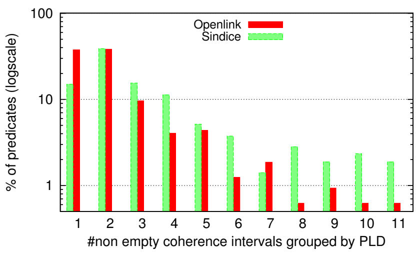
There are other alternatives to measure coherence for a store [21]. Since we could not empirically observe significant differences among the different measures discussed in [21], we use the result-based coherence measure in this paper.

For the two stores under analysis, Figure 4(a) illustrates the number of predicates that fall into different intervals of coherence values; the y-axis is in logarithmic scale, and the linear x-axis intervals represent the coherence measures as percentages (the right of the graph indicates increasingly coherent predicates). The figure shows that the OpenLink endpoint is more in sync with current Web data than Sindice; we believe that OpenLink was extensively updated in Feb. 2012. We measured that 67% of the tested predicates in the OpenLink index are entirely up-to-date ($\mathsf{coh}_r(p) = 1$), versus 30% of the predicates for the

[5] The statistics took over a week to collect politely (5 s delay). Maintaining coherence statistics is non-trivial, but out of scope. Discussion can rather be found in [21, § 4].



(a) Coherence distribution



(b) PLD coherence variations

Fig. 4. Distribution of predicate coherence values and variation across PLDs

Sindice endpoint. In contrast, information for 14% of the tested predicates in the OpenLink index are entirely missing or out-of-date ($\mathsf{coh}_r(p) = 0$), versus 40% for Sindice; these high percentages are due to partial coverage of Web sources, outdated data-dumps in the index, and predicates with dynamic values.

In more detail, Table 1 shows the top 5 predicates where $\mathsf{coh}_r(p) = 0$ for both stores, ordered by the number of queries in which they featured as a result. First, we see a mix of dynamic time-stamp predicates that change for every access or modification of a document (`swivt:creationDate`, `swivt:wikiPageModificationDate` and `aims:hasDateCreated`). Second, we see predicates not covered by the index. For Sindice, the incoherent `*:doi` predicates are due to a lack of coverage of the `dx.doi.org` domain and the high incoherency of `skos:` predicates is due to bulk changes in the `esd-toolkit.eu`, `esd.org.uk` and `bio2rdf.org` domains; for OpenLink, the incoherency of `vitro:mostSpecificType` relates to data on the `cornell.edu` domain.

We further analysed the correlation for coherence estimates of the same predicates *across* the two stores. We used Kendall's τ, which measures the agreement in ordering for two measures in a range of $[-1, 1]$, where -1 indicates perfectly inverted ordering and 1 indicates the exact same ordering. The τ-score across the two stores was 0.16, with a negligible p-value, indicating a weak, significant and positive correlation between the coherence of predicates for the two stores. The low correlation highlights the store-specific nature of these measures, which are as much about index coverage than about the dynamicity of values. As such, our approach tackles both the global problem of dynamicity and the central problem of store coverage.

Finally, we looked at the correlation between the selectivity of predicates (i.e., how often they occur) and their coherence, which may have potential consequences for query planning. Specifically, for each store, we compared the number of (Web) results generated for each predicate across all queries and their $\mathsf{coh}_r(p)$ value. The τ-value for OpenLink was 0.1, indicating that less selective patterns tend to have slightly lower coherence; the analogous τ-value for Sindice

Table 1. Most dynamic and prevalent predicates

№	OpenLink		Sindice	
	pred.	*queries*	*pred.*	*queries*
1	`swivt:creationDate`	510	`swivt:creationDate`	118
2	`vitro:mostSpecificType`	104	`skos:narrower`	48
3	`swivt:wikiPageModificationDate`	45	`skos:historyNote`	43
4	`aims:hasDateCreated`	42	`bibo:doi`	34
5	`madsrdf:hasCloseExternalAuthority`	31	`prism21:doi`	34

was -0.03, indicating a *very* slight correlation in the opposite direction. Though limited, we take this as anecdotal evidence to suggest that correlation between the selectivity and coherence of predicates is weak, if any.

Above, we naïvely assume a single coherence value for predicates in all cases, ignoring subject or object URIs: keeping information for each subject/object would have a high overhead. However, we can generalise subject/object values into pay-level-domains (PLD)[6] and then track coherence for predicate–domain pairs. Thus, we mapped the entity URIs of the queries to their PLDs (581 PLDs with a maximum of 74 queries per domain) and resolved the coherence of predicates for individual PLDs. Focussing on the coherence measure, we divided the scores into eleven intervals as per the x-axis of Figure 4(b), and for each predicate, count how many intervals it falls into for different PLDs. We observe that the subject and object URIs can be ignored for roughly 40% of the OpenLink and roughly 15% for the Sindice predicates. However, we see the importance of tracking coherence for predicate–domain pairs for the remaining predicates. The plurality of predicates (~40%) show two intervals of coherence values.

5 Query Planner

Our hybrid engine combines centralised and decentralised/live query execution to obtain a balance between fresh and fast results. The QUERY PLANNER is responsible for splitting the query into a part for execution against the centralised store and a part for live execution. We focus on evaluating conjunctive queries (i.e., SPARQL BGPs). Other SPARQL features—except `OPTIONAL` (and `MINUS` & [`NOT`] `EXISTS` in SPARQL 1.1) for which LTBQE is ill-suited since it (typically) does not operate over a pre-defined dataset—can be layered on top.

As indicated before, we argue for a single split of the query plan into one "central" (executed by the store) and one live part. While in theory it would also be possible to use multiple splits, the resulting intertwined dependencies between the central and live parts would lead to very complex query planning, and would require shipping bindings back and forth between the two engines. Thus, advanced splitting approaches are better suited to controlled environments (i.e., not public endpoints). We discuss this issue in more detail in [21].

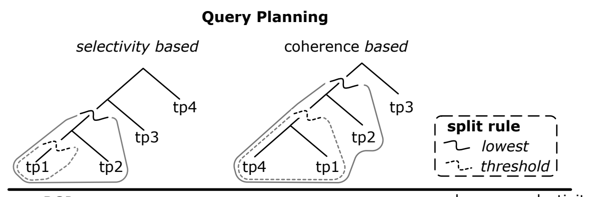
[6] A pay-level-domain is the domain name one has to register and pay for.

Given the focus on a split into (at most) two parts, the results of the first part serve as input bindings for the second part. We must then decide whether the central part is processed first (i.e., at the bottom of the query plan) or last (i.e., at the top of the plan). As previously discussed in [21], there are constraints inherent to live SPARQL query execution methods, such as the need for dereferenceable URIs in some of the query patterns. By running the central part first, we do not only obtain the store's results more quickly, but also provide additional dereferenceable URI bindings for the live querying phase (passed to the live engine using the `VALUES` [previously `BINDINGS`] clause in SPARQL 1.1).

As per traditional database query-planning approaches, we must then decide the execution order of (commutative) join patterns. However, instead of only optimising for speed, we now *also* wish to optimise for freshness. An intuitive approach, which we call **coherence-based ordering**, is to build a query plan with the most coherent patterns at the bottom for central execution, and the most incoherent patterns at the top for live execution. This increases the likelihood that the final result set is fresh and it limits the number of patterns executed live. However, the most coherent patterns may also be the least *selective* (i.e., return the most bindings) thus inflating the number of intermediate results to process. Consequently, this can increase the number of bindings for the patterns executed live, potentially hurting the performance. Because of this and backed by the absence of correlation between coherence and selectivity, we also consider another approach following traditional **selectivity-based reordering**, where the most restrictive patterns are executed first reducing intermediate results.

After selecting an ordering, we must also select a *split pattern*. The split pattern is the position in the query plan in which the query is divided into the two parts. Everything below the split is executed centrally, and everything above and including the split pattern is executed live. Following the same intuition of executing low-coherence patterns live, one option is to choose the **most incoherent** triple pattern as the split. However, the central store may still receive highly incoherent patterns (below the max) for which it will return incoherent results. Another approach is to define a constant value indicating a **threshold of incoherence**, where the lowest pattern breaching the threshold becomes the split pattern; this ensures that the store does not receive patterns that are highly incoherent. A further option is to split by a **fixed position** n, whereby the n bottom patterns are executed by the store and the rest are run live. Choosing between the different split options affects the core trade-off of fresh vs. fast results, and thus could be parameterised for individual user needs.

Figure 5 depicts examples of hybrid query plans, including our two different choices of ordering, as well as some possible split choices. In the selectivity-based ordering, we see that different types of coherence thresholds may lead to more patterns being run live than when explicitly ordered by coherence. Conversely, at the base of the plan for the coherence-based ordering, we see that **tp4** will return a lot of intermediate bindings (since it has a low selectivity) and does not share a join variable with **tp1** on the right-hand side of the join. In general, one may expect that the selectivity-based operator order would provide low answer times by minimising intermediate bindings, but would return less fresh results since



BGP	coherence	selectivity
tp1: <*http://status.samnoble.org/user/1*> foaf:knows ?o1 .	0.86	0.98
tp2: ?o1 foaf:interest ?o2 .	0.32	0.43
tp3: ?o1 swivt:wikiPageModificationDate ?o3 .	0.00	0.21
tp4: ?o2 foaf:accountProfilePage ?o0 .	0.91	0.15

Fig. 5. Example hybrid query plans for different orderings and splits

low coherence patterns can appear below the split. However, this ordering also ends up pushing more patterns live since patterns with low selectivity and high coherence are often above the split. Conversely, a coherence-based ordering will lead to more intermediate results, but will run more patterns centrally. Thus, a general conclusion about which ordering is preferable is not possible; we instead compare combinations of orderings and splits on an empirical basis in Section 6.

In practice, for selectivity ordering, we create our hybrid SPARQL query plan using ARQ based on a "variable counting" technique [14], which estimates the selectivity of different triple patterns based on rules involving the number and position of variables it contains. This could be replaced with cost-based planning using empirical selectivity estimates, but we would need statistics about the underlying data. Thus, following a rule-based approach is more in line with targets (T3) system independence, and (T4) lightweight implementation (cf. Section 3). A coherence-based operator order is supported by reordering the triple patterns in the query plan produced by ARQ based on their coherence values.

In fact, since we consider the store and the live-query component as black boxes, both sub-queries will be reordered by the respective engines, thus mitigating some of the performance penalty associated with the possibly naïve ordering used to decide the split in the hybrid query plan. For example, referring back to the coherence-ordered plan of Figure 5, if the lowest coherence split rule is applied, the store may internally decide to run **tp1**, **tp2** and then **tp4** in that order, avoiding the (huge) expense of running **tp4** first.

6 Evaluation

We now evaluate our proposed hybrid query execution. Our concrete goals are as follows: (i) to prove concept in a realistic setting and show that with the correct plan, hybrid query execution can extend and freshen up central results while speeding up live results; (ii) to evaluate different query plan strategies by comparing (ii_a) selectivity- and coherence-based ordering and (ii_b) different split strategies for the query planning. In parallel, we are interested to see how useful our coherence estimates are for the hybrid-query planning phase.

To do so, our evaluation is run against the two selected public endpoints: Sindice and OpenLink. For this, we require a set of evaluation queries that are answerable by a Linked Data query engine. We would like these queries to have broad coverage of diverse Web sources in order to properly test coherence estimates and hybrid splits. Hence, we generate queries from the Billion Triple Challenge 2011 dataset, which covers a broad range of Web documents. We apply a random walk technique on the dataset, selecting random paths between dereferenceable URIs in the data to produce queries that will give non-empty results if executed with the LTBQE LIVE QUERY INTERFACE (see [19] for more details). Using this method, we produce 200 SPARQL `SELECT` queries of different shapes (star, path, mixed), with varying numbers of patterns (2–6), randomly assigned distinguished variables, and at least one pattern above and below a coherence threshold of 0.5 (i.e., suitable for hybrid execution). After filtering out queries that produce empty results (e.g., due to offline sources) or result in endpoint errors (like timeouts), we obtained a set of 98 stable queries for the OpenLink store and 91 stable queries for the Sindice store. We reran all experiments four times over a period of eight days to verify repeatability.

To evaluate different orders and different cut-off positions, we created hybrid query plans for each query using both the selectivity- and coherence-based reordering strategies. Each query plan is then run entirely live, entirely by the store, and also run for every possible split position in both orders where part goes live and part goes to the store. We execute all split positions for simple convenience: we can then later compare different *a priori* strategies for picking a single split by looking up the corresponding results (and not rerunning queries).

In terms of the repeatability of results, for each configuration, we measured deviations for recall of results and query time across the four runs vs. the best approach (highest recall, lowest time). We then calculated the mean of these absolute deviations across all queries. For the live (LTBQE) execution, we measured a recall deviation of 3% and a time deviation of 2.7%. For the various other configurations, the recall deviation varied between 0–5% for OpenLink and between 0–2% for Sindice. Although the recall of the stores was very stable, we observed average time deviations of up to 36% for OpenLink and 17% for Sindice, indicating variable query response times. Acknowledging that public endpoints and remote data sources can be unstable, we wish to factor out this instability to derive comparable results across different hybrid strategies (we wish to compare different hybrid query plans, not the performance of public SPARQL endpoints). Thus, to avoid outliers, for each query and each configuration, we only select the best run in terms of recall, and in case two runs have the same recall, we select the one with the lower query time.

We first focus on ordering. To initially prove concept, we want to show that, in practice, hybrid query execution can *potentially* improve the freshness of results vs. the store while reducing query time vs. live querying. Table 2 presents such an analysis for both stores, where we see the potential percentage of queries that can be improved using our hybrid approach for both orders assuming (for the moment) that the best possible split position is picked (i.e., given the results, we select the split position that gave the highest recall and if tied, the lowest time; we evaluate split-selection strategies later). Recall is measured relative to the live results, which we know to be fresh. For OpenLink, we see that the recall of

Table 2. For both stores, the percentage of queries that can potentially be improved for each order assuming the best split position is picked

improvement	**OpenLink**		**Sindice**	
	`sel`	`coh`	`sel`	`coh`
Better Than Centralised Recall:	43%	53%	87%	91%
Better Than or Equal Centralised Recall:	97%	100%	99%	97%
Better Than Live Time:	92%	45%	16%	3%
Better Than Centralised Recall & Live Time	39%	13%	8%	1%
Better Than Or Equal Centralised Recall & Live Time:	92%	45%	16%	3%

the store can only be improved for roughly half of the queries; however, the recall of the store is already 1 in many cases and cannot be improved, only equalled. Equal ties in time are much more rare. In terms of improving the time for live results, the `sel` ordering seems much more beneficial for OpenLink than `coh`, likely due to fewer intermediate results being generated in the former ordering: `sel` improves or equals the store's recall while improving the live time in 92% of the queries. For Sindice, we found that the store often returned no query results: 84% of the queries ran entirely live as a fallback. Thus, the recall of many queries can be improved outright, but few queries are faster than the live approach. Since only 16% of the queries for Sindice are run in a truly hybrid fashion, we henceforth focus on OpenLink—all hybrid query results for Sindice were very close to the live approach. Table 2 shows that, in an ideal case, the hybrid approach can indeed improve result freshness while reducing the time required to process queries. Furthermore, the chosen ordering strategy seems to have a clear impact on freshness and query time.

We now compare concrete split strategies that *a priori* select a split based on the query and coherence/selectivity estimates (i.e., as would be required for hybrid query planning in reality, where an ordering and a split strategy are sufficient to generate a plan). We also look at the *degree* to which central recall is improved and live querying is sped up. To compare different splits and ordering combinations, we first filter out queries that, across the four runs, did not provide results for all possible split positions and orderings for one or more of the setups. We also removed queries with only two patterns, for which the choice of split is trivial. This results in a final set of 43 queries.[7]

For each ordering and for a variety of different split strategies, Figure 6 plots the aggregate speed up and recall ratio versus live querying. Specifically, to calculate speed up, the total time taken by the live approach to run all queries is divided by the total time taken for each individual approach; e.g., a speed up of 6 indicates that the approach in question was 6× faster than live querying. Conversely, recall is measured by taking live querying as the gold standard. Live querying is thus placed at $(1, 1)$. Orderings are intuitively represented by `sel-*` and `coh-*`. The best split approaches are represented by `*-best`; these splits cannot be determined before query execution, but rather represent the ideal case. Splitting at the most incoherent pattern is indicated by `*-incoh`. Using a

[7] Queries are available online at the following address:
http://code.google.com/p/lidaq/source/browse/queries/iswc-2012.tar.gz

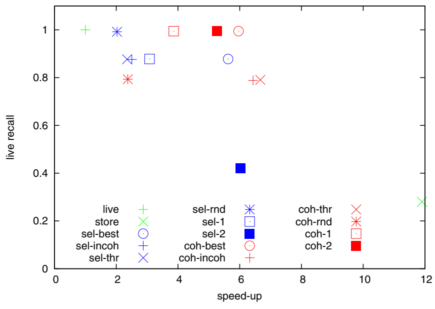


Fig. 6. Recall vs. speed-up trade-off for different hybrid plans (`thr` = 0.5)

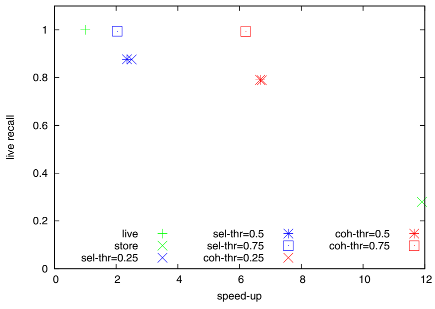


Fig. 7. Recall vs. speed-up trade-off for different thresholds

coherence threshold of 0.5 to perform the split is indicated by `*-thr`. A random split (i.e., a guess) is indicated by `*-rnd`. Fixed split positions are indicated by `*-1` and `*-2` for $n = 1, 2$. Note that the threshold strategies `*-incoh`/`*-thr` can go fully live or fully central depending on the coherence values found for the query, whereas `*-best`, `*-rnd`, `*-1` and `*-2` must split the query. Figure 7 shows the same analysis, but for varying coherence threshold values.

In fact, both plots offer an empirical version of the trade-off introduced in Figure 1, where our hybrid strategies sit between live querying and the store. For both graphs, we see that the store is the fastest, and about 12× faster than the live approach; however, recall is poor. Perhaps the best hybrid approach is `coh-thr=0.75` in Figure 7, which maintains an almost perfect recall but offers a speed up of more than 6× live querying, slightly beating the ideal of `coh-best` (which must split the query). Interestingly, some fixed-position split strategies—particularly `coh-2` in Figure 6, which is ∼ 5× faster than live, but maintains an almost perfect recall—can approach the ideal of `coh-best` quite closely.

While such an aggregated view presents high-level insights into the overall performance of the different strategies, we cannot identify the distribution over the queries in terms of achieved freshness and time. This is supported by Figure 8 and Figure 9, which show the recall for each query and the query time for each query respectively. We plot the number of queries on the x-axis that achieve a certain recall or time ratio shown on the y-axis. All 43 queries are sorted for each approach separately, providing a global view on each performance, but not supporting a per-query comparison of the strategies. While querying the store results in the fewest queries with a recall of 1, it also results in the most queries with a recall between 0 and 1. On the contrary, `coh-best` and `coh-2` are tied for keeping 100% recall across 42 queries and provide 0% only for the last query. Interestingly, most queries run with any hybrid strategy result in a recall of either 1 or 0. As expected, Figure 9 shows that query times for the centralised store are far below all other approaches in most cases. However, it is in fact slower for 2 anomalous queries that OpenLink struggles with.[8] Though all hybrid strategies were as fast or faster than live querying ($y = 1$) in all cases,

[8] One such example at the time of writing was `http://bit.ly/IPRec9`

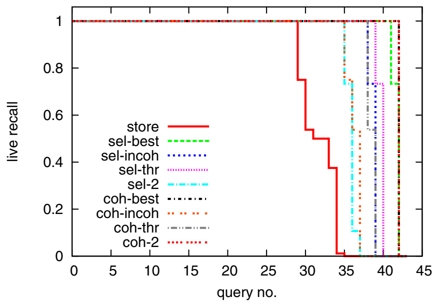


Fig. 8. Queries ordered by *recall* for different order and split strategies

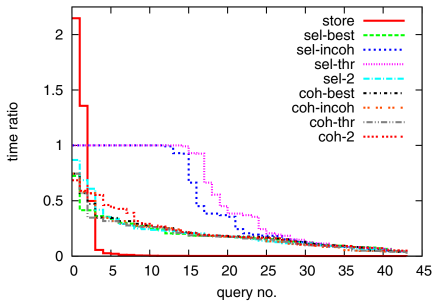


Fig. 9. Queries ordered by *time* for different order and split strategies

we see that `sel-incoh` and `sel-thres` did not show a time improvement for around 13 queries, where they were likely run completely live.

Eventually, we are interested in how recall and query times compare for different approaches on the same queries. Thus, in Figure 10 and Figure 11 we ordered the queries for each approach identically (using the results of the store for ordering). This means that in these figures each point on the x-axis presents the *same* query for each approach. In this case, plotting the absolute values would not allow any meaningful insights due to the ups and downs that each plot would show. Instead, we plot an "evolving average", whereby the result for query n indicates the average value for all queries up to and including n. This allows to compare the degree of increase or decrease in recall and time at each point, i.e., for each query. Interestingly, we see that the store can sometimes return better recall than the hybrid approaches, as happens for query 27, where the interim bindings returned by OpenLink cannot be dereferenced. However, the recall is improved by the hybrid approaches for the subsequent queries. It is further interesting to observe that the hybrid approaches seem to be "grouped", i.e., `coh-best`, `coh-2` and `sel-best` show very similar performance over all queries, while `coh-incoh` first "follows" `sel-thr` and others, but performs similar to `sel-2` for later queries. The evolving average of the query times in Figure 11 basically confirm the results from Figure 9.

In summary, we found that sending more patterns live with fewer bindings (as with the selectivity-based ordering) is in parts faster than sending fewer patterns live with more bindings (coherence-based ordering). Though selectivity *ordering* does not consider coherence, a coherence-based split ensures that more of the query is executed live to compensate. The lower query times suggested by the coherence-based orderings are often compensated by the low selectivities of operators in the lower levels of query plans. Still, as a guideline, if the objective is to maximise recall, one should choose coherence-based ordering, which is still faster than the live approach. If one is willing to sacrifice some recall for even faster results, then selectivity-based ordering is a good choice. However, the performance of the selectivity-based approaches seems to heavily depend on the actually chosen split strategy. Generally, the question of how to pick the

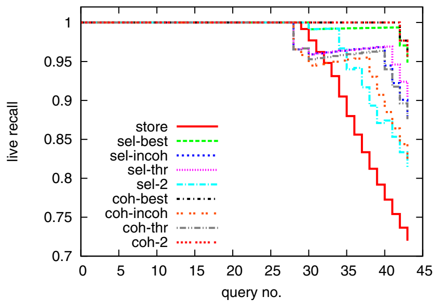


Fig. 10. Evolving average for *query recall* with different order and split strategies

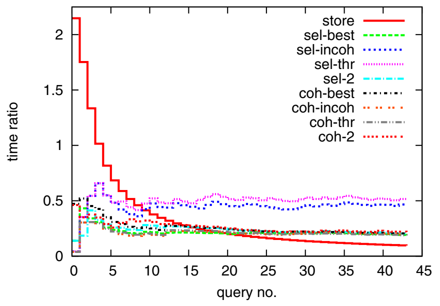


Fig. 11. Evolving average for *query time* with different order and split strategies

split position cannot be ultimately answered without taking the actual query and other characteristics into account. While the actual value of the coherence threshold did not have an impact as high as expected, we could show that the coherence estimates themselves are of great benefit, especially for ordering.

7 Conclusion

Based on an empirical study showing that popular public SPARQL stores struggle to maintain coherent cached indexes of Linked Data, we propose a hybrid query architecture that aims to combine the best from centralised indexes and novel live querying approaches. We discussed extracting coherence measures from centralised endpoints based on probing queries, and showed that they can be combined with reordering and hybrid-split strategies to design an effective hybrid query plan that speeds up live results while freshening up centralised results.

Still, some open questions remain. We have looked at a wide variety of configurations, which hint at the potential complexity of hybrid query planning. More complex cost models—including, e.g., the potential for multiple splits—may reveal novel optimisations that we have not considered herein, further pushing the boundaries of fresh vs. fast results. Furthermore, we can only estimate the accuracy of store results using coherence estimates; other mechanisms that cross-check the sources of data (i.e., the named graphs from which the store computes answers) against their current versions could yield yet more accurate statistics. Also, we consider the live and centralised query components to be strongly decoupled. However, the store may serve as a source selection index to enhance the live results given by a zero-knowledge approach such as LTBQE.

Given the potential scope and dynamicity of Linked Data, query engines will need to employ a range of techniques to efficiently offer fresh results with broad coverage. We believe that our hybrid query approach makes a significant step in this direction by combining results from centralised and decentralised query engines. Still, we may only have scratched the surface of what is possible.

References

1. Buil-Aranda, C., Arenas, M., Corcho, O.: Semantics and Optimization of the SPARQL 1.1 Federation Extension. 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. 1–15. Springer, Heidelberg (2011)
2. Bishop, B., Kiryakov, A., Ognyanoff, D., Peikov, I., Tashev, Z., Velkov, R.: FactForge: A fast track to the web of data. SWJ 2(2), 157–166 (2011)
3. Bizer, C., Jentzsch, A., Cyganiak, R.: State of the LOD Cloud (v0.3). Online report, FUB/DERI (2011)
4. Erling, O., Mikhailov, I.: RDF Support in the Virtuoso DBMS. In: Pellegrini, T., Auer, S., Tochtermann, K., Schaffert, S. (eds.) Networked Knowledge - Networked Media. SCI, vol. 221, pp. 7–24. Springer, Heidelberg (2009)
5. Görlitz, O., Staab, S.: Federated Data Management and Query Optimization for Linked Open Data. In: Vakali, A., Jain, L.C. (eds.) New Directions in Web Data Management 1. SCI, vol. 331, pp. 109–137. Springer, Heidelberg (2011)
6. 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)
7. Karnstedt, M., Sattler, K., Geist, I., Höpfner, H.: Semantic Caching in Ontology-based Mediator Systems. In: Berliner XML-Tage. XML-Clearinghouse (2003)
8. 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)
9. Ladwig, G., Tran, T.: SIHJoin: Querying Remote and Local Linked Data. 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. 139–153. Springer, Heidelberg (2011)
10. 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)
11. Podlipnig, S., Böszörményi, L.: A survey of Web cache replacement strategies. ACM Comput. Surv. 35(4), 374–398 (2003)
12. 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)
13. 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)
14. Stocker, M., Seaborne, A.: ARQo: The architecture for an ARQ static query optimizer. Technical report, HP Labs Bristol (2007)
15. Stuckenschmidt, H., Vdovjak, R., Houben, G.-J., Broekstra, J.: Index structures and algorithms for querying distributed RDF repositories. In: WWW. ACM (2004)
16. Tran, T., Zhang, L., Studer, R.: Summary Models for Routing Keywords to Linked 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. 781–797. Springer, Heidelberg (2010)

17. Tummarello, G., Delbru, R., Oren, E.: Sindice.com: Weaving the Open Linked Data. 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.) ISWC/ASWC 2007. LNCS, vol. 4825, pp. 552–565. Springer, Heidelberg (2007)
18. Umbrich, J., Hausenblas, M., Hogan, A., Polleres, A., Decker, S.: Towards dataset dynamics: Change frequency of linked open data sources. In: LDOW. CEUR (2010)
19. Umbrich, J., Hogan, A., Polleres, A., Decker, S.: Improving the Recall of Live Linked Data Querying through Reasoning. In: Krötzsch, M., Straccia, U. (eds.) RR 2012. LNCS, vol. 7497, pp. 188–204. Springer, Heidelberg (2012)
20. Umbrich, J., Hose, K., Karnstedt, M., Harth, A., Polleres, A.: Comparing data summaries for processing live queries over Linked Data. WWWJ 14(5-6), 495–544 (2011)
21. Umbrich, J., Karnstedt, M., Hogan, A., Parreira, J.X.: Freshening up while Staying Fast: Towards Hybrid SPARQL Queries. 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. 164–174. Springer, Heidelberg (2012)
22. Umbrich, J., Karnstedt, M., Parreira, J.X., Polleres, A., Hauswirth, M.: Linked Data and live querying for enabling support platforms for Web dataspaces. In: DESWEB Workshop, ICDE. IEEE Computer Society (2012)
23. Williams, G.T., Weaver, J.: Enabling Fine-Grained HTTP Caching of SPARQL Query Results. 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. 762–777. Springer, Heidelberg (2011)

Provenance for SPARQL Queries

Carlos Viegas Damásio[1], Anastasia Analyti[2], and Grigoris Antoniou[3]

[1] CENTRIA, Departamento de Informática Faculdade de Ciências e Tecnologia
Universidade Nova de Lisboa, 2829-516 Caparica, Portugal
cd@fct.unl.pt

[2] Institute of Computer Science, FORTH-ICS, Crete, Greece
analyti@ics.forth.gr

[3] Institute of Computer Science, FORTH-ICS, and
Department of Computer Science, University of Crete, Crete, Greece
antoniou@ics.forth.gr

Abstract. Determining trust of data available in the Semantic Web is fundamental for applications and users, in particular for linked open data obtained from SPARQL endpoints. There exist several proposals in the literature to annotate SPARQL query results with values from abstract models, adapting the seminal works on provenance for annotated relational databases. We provide an approach capable of providing provenance information for a large and significant fragment of SPARQL 1.1, including for the first time the major non-monotonic constructs under multiset semantics. The approach is based on the translation of SPARQL into relational queries over annotated relations with values of the most general m-semiring, and in this way also refuting a claim in the literature that the `OPTIONAL` construct of SPARQL cannot be captured appropriately with the known abstract models.

Keywords: How-provenance, SPARQL queries, m-semirings, difference.

1 Introduction

A general data model for annotated relations has been introduced in [9], for positive relational algebra (i.e. excluding the difference operator). These annotations can be used to check derivability of a tuple, lineage, and provenance, perform query evaluation of incomplete database, etc. The main concept is the notion of $\mathcal{K}$-relations where tuples are annotated with values (tags) of a commutative semiring $\mathcal{K}$, while positive relational algebra operators semantics are extended and captured by corresponding compositional operations over $\mathcal{K}$. The obtained algebra on $\mathcal{K}$-relations is expressive enough to capture different kinds of annotations with set or bag semantics, and the authors show that the semiring of polynomials with integer coefficients is the most general semiring. This means that to evaluate queries for any positive algebra query on an arbitrary semiring, one can evaluate the query in the semiring of polynomials (factorization property of [9]). This work has been extended to the case of full relational algebra in [8] by considering the notion of semirings with a monus operation (m-semirings [2]) and constant annotations, and the factorization property is proved for the special m-semiring that we denote by $\mathcal{K}_{dprovd}$.

The use of these abstract models based on $\mathcal{K}$-relations to express provenance in the Semantic Web has been advocated in [12]. However, the authors claim that the existing m-semirings are not capable to capture the appropriate provenance information for SPARQL queries. This claim is supported by the authors using a simple example, which we have adapted to motivate our work:

Example 1. Consider the following RDF graph expressing information about users' accounts and homepages, resorting to the FOAF vocabulary:

```
@prefix people: <http://people/> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
people:david foaf:account <http://bank> .
people:felix foaf:account <http://games> .
<http://bank> foaf:accountServiceHomepage <http://bank/yourmoney>.
```

The SPARQL query

```
PREFIX foaf <http://xmlns.com/foaf/0.1/>
SELECT *
WHERE { ?who foaf:account ?acc .
        OPTIONAL { ?acc foaf:accountServiceHomepage ?home }
}
```

returns the solutions (mappings of variables):

?who	?acc	?home
<http://people/david>	<http://bank>	<http://bank/yourmoney>
<http://people/felix>	<http://games>	

However, if the last triple is absent from the graph then the solutions are instead:

?who	?acc	?home
<http://people/david>	<http://bank>	
<http://people/felix>	<http://games>	

In order to track provenance of data, each tuple of data can be tagged with an annotation of a semiring. This annotation can be a boolean, e.g. to annotate that the tuple is trusted or not, a set of identifiers of tuples returning lineage of the tuple, or more complex annotations like the polynomials semiring to track full how-provenance [9,8], i.e. how a tuple is generated in the result under bag semantics.

Returning to the introductory example, assume that we represent the 3 triples in the input RDF graph as the ternary K_{dprovd}-relation (with the obvious abbreviations), where the last column contains the triple identifier (annotation):

Triples

sub	pred	obj	
<david>	<account>	<bank>	t_1
<felix>	<account>	<games>	t_2
<bank>	<accountServiceHomepage>	<bank/yourmoney>	t_3

The expected annotation of the first solution of the SPARQL query is $t_1 \times t_3$, meaning that the solution was obtained by joining triples identified by t_1 and t_3, while for the second solution the corresponding annotation is simply t_2. However, if we remove the last tuple we obtain a different solution for `david` with annotation just t_1. The authors in [12] explain why the existing approaches to provenance for the Semantic Web cannot handle the situation of Example 1, basically because there are different bindings of variables depending on the absence/presence of triples, and it is claimed that the m-semiring K_{dprovd} also cannot handle it. The rest of our paper shows that this last claim is wrong, but that requires some hard work and long definitions since the method proposed relies on the translation of SPARQL queries into relational algebra. The result is the first approach that provides adequate provenance information for `OPTIONAL`, `MINUS` and `NOT EXISTS` constructs under the multiset (bag) semantics of SPARQL.

The organization of the paper is the following. We review briefly in the next section the basics of K-relations. The SPARQL semantics is introduced in Section 3, and its translation into relational algebra is the core of the paper and can be found in Section 4. Using the relational algebra translation of SPARQL, we use K_{dprovd} to annotate SPARQL queries and show in Section 5 that Example 1 is properly handled. We finish with some comparisons and conclusions.

2 Provenance for *K*-Relations

A commutative semiring is an algebraic structure $\mathcal{K} = (\mathbb{K}, \oplus, \otimes, 0, 1)$ where $(\mathbb{K}, \oplus, 0)$ is a commutative monoid ($\oplus$ is associative and commutative) with identity element 0, $(\mathbb{K}, \otimes, 1)$ is a commutative monoid with identity element 1, the operation $\otimes$ distributes over $\oplus$, and 0 is the annihilating element of $\otimes$. In general, a tuple is a function $t : U \to \mathbb{D}$ where U is a finite set of attributes and $\mathbb{D}$ is the domain of values, which is assumed to be fixed. The set of all such tuples is U-`Tup` and usual relations are subsets of U-`Tup`. A $\mathcal{K}$-relation over U is a function $R : U\text{-}\mathtt{Tup} \to \mathbb{K}$, and its support is $supp(R) = \{t \mid R(t) \neq 0\}$.

In order to cover the full relational operators, the authors in [8] assume that the $\mathcal{K}$ semiring is naturally ordered (i.e. binary relation $x \preceq y$ is a partial order, where $x \preceq y$ iff there exists $z \in \mathbb{K}$ such that $x \oplus z = y$), and require additionally that for every pair x and y there is a least z such that $x \preceq y \oplus z$, defining in this way $x \ominus y$ to be such smallest z. A $\mathcal{K}$ semiring with such a monus operator is designated by m-semiring. Moreover, in order to capture duplicate elimination, the authors assume that the m-semiring is finitely generated. The query language[1] $\mathcal{RA}^+_{\mathcal{K}}(-, \delta)$ has the following operators [8]:

empty relation: For any set of attributes U, we have $\emptyset : U\text{-}\mathtt{Tup} \to \mathbb{K}$ such that $\emptyset(t) = 0$ for any t.

union: If $R_1, R_2 : U\text{-}\mathtt{Tup} \to \mathbb{K}$ then $R_1 \cup R_2 : U\text{-}\mathtt{Tup} \to \mathbb{K}$ is defined by: $(R_1 \cup R_2)(t) = R_1(t) \oplus R_2(t)$.

[1] The authors use instead the notation $\mathcal{RA}^+_{\mathcal{K}}(\setminus, \delta)$.

projection: If $R : U\text{-}\mathtt{Tup} \to \mathbb{K}$ and $V \subseteq U$ then $\Pi_V(R) : V\text{-}\mathtt{Tup} \to \mathbb{K}$ is defined by $(\Pi_V(R))\,(t) = \bigoplus_{t=t' \text{ on } V \text{ and } R(t')\neq 0} R(t')$.

selection: If $R : U\text{-}\mathtt{Tup} \to \mathbb{K}$ and the selection predicate P maps each U-tuple to either 0 or 1 depending on the (in-)equality of attribute values, then $\sigma_P(R) : U\text{-}\mathtt{Tup} \to \mathbb{K}$ is defined by $(\sigma_P(R))\,(t) = R(t) \otimes P(t)$.

natural join: If $R_i : U_i\text{-}\mathtt{Tup} \to \mathbb{K}$, for $i = 1, 2$, then $R_1 \bowtie R_2$ is the $\mathcal{K}$-relation over $U_1 \cup U_2$ defined by $(R_1 \bowtie R_2)(t) = R_1(t) \otimes R_2(t)$.

renaming: If $R : U\text{-}\mathtt{Tup} \to \mathbb{K}$ and $\beta : U \to U'$ is a bijection then $\rho_\beta(R)$ is the $\mathcal{K}$-relation over U' defined by $(\rho_\beta(R))(t) = R(t \circ \beta^{-1})$.

difference: If $R_1, R_2 : U\text{-}\mathtt{Tup} \to \mathbb{K}$ then $R_1 - R_2 : U\text{-}\mathtt{Tup} \to \mathbb{K}$ is defined by: $(R_1 - R_2)(t) = R_1(t) \ominus R_2(t)$.

constant annotation: If $R : U\text{-}\mathtt{Tup} \to \mathbb{K}$ and k_i is a generator of $\mathbb{K}$ then $\delta_{k_i} : U\text{-}\mathtt{Tup} \to \mathbb{K}$ is defined by $(\delta_{k_i}(R))(t) = k_i$ for each $t \in \mathit{supp}(R)$ and $(\delta_{k_i}(R))(t) = 0$ otherwise.

One major result of [8] is that the factorization property can be obtained for $\mathcal{RA}^+_\mathcal{K}(-,\delta)$ by using a special m-semiring with constant annotations that we designate by $\mathcal{K}_{dprovd}$. $\mathcal{K}_{dprovd}$ is the free m-semiring over the set of source tuple ids X, which is a free algebra generated by the set of (tuple) identifiers in the equational variety of m-semirings. Elements of $\mathcal{K}_{dprovd}$ are therefore terms defined inductively as: identifiers in X, 0, and 1 are terms; if s and t are terms then $(s + t)$, $(s \times t)$, $(s - t)$, and $\delta_{k_i}(t)$ are terms, and nothing else is a term. In fact annotations of $\mathcal{K}_{dprovd}$ are elements of the quotient structure of the free terms with respect to the congruence relation induced by the axiomatization of the m-semirings, in order to guarantee the factorization property (see [8] for more details). In our approach, X will be the set of graph and tuple identifiers.

We slightly extend the projection operator, by introducing new attributes whose value can be computed from the other attributes. In our approach, this is simply syntactic sugar since the functions we use are either constants or return one of the values in the arguments.

3 SPARQL Semantics

The current draft of SPARQL 1.1 [1] defines the semantics of SPARQL queries via a translation into SPARQL algebra operators, which are then evaluated with respect to a given RDF dataset. In this section, we overview an important fragment corresponding to an extension of the work in [10] that presents the formal semantics of the first version of SPARQL. The aim of our paper is on the treatment of non-monotonic constructs of SPARQL, namely OPTIONAL, MINUS and NOT EXISTS, and thus we focus in the SELECT query form, ignoring property paths, GROUP graph patterns and aggregations, as well as solution modifiers. The extension of our work to consider all the graph patterns is direct from the results presented. Regarding FILTER expressions, we analyse with detail the EXISTS and NOT EXISTS constructs, requiring special treatment. We assume the reader has basic knowledge of RDF and we follow closely the presentation of [1]. For more details the reader is referred to sections 17 and 18 of the current SPARQL 1.1 W3C working draft. We also make some simplifying assumptions that do not affect the results of our paper.

3.1 Basics

Consider disjoint sets of IRI (absolute) references $\mathbf{I}$, blank nodes $\mathbf{B}$, and literals $\mathbf{L}$ including plain literals and typed literals, and an infinite set of variables $\mathbf{V}$. The set of RDF terms is $\mathbf{T} = \mathbf{IBL} = \mathbf{I} \cup \mathbf{B} \cup \mathbf{L}$. A triple[2] $\tau = (s, p, o)$ is an element of $\mathbf{IBL} \times \mathbf{I} \times \mathbf{IBL}$ and a graph is a set of triples. Queries are evaluated with respect to a given RDF Dataset $D = \{G, (<\mathtt{u_1}>, G_1), (<\mathtt{u_2}>, G_2), \ldots, (<\mathtt{u_n}>, G_n)\}$, where G is the default graph, and each pair $(<\mathtt{u_i}>, G_i)$ is called a named graph, with each IRI $\mathtt{u_i}$ distinct in the RDF dataset, and G_i being a graph.

3.2 Graph Patterns

SPARQL queries are defined by graph patterns, which are obtained by combining triple patterns with operators. SPARQL graph patterns are defined recursively as follows:

– The empty graph pattern ().
– A tuple $(\mathbf{IL} \cup \mathbf{V}) \times (\mathbf{I} \cup \mathbf{V}) \times (\mathbf{IL} \cup \mathbf{V})$ is a graph pattern called triple pattern[3];
– If P_1 and P_2 are graph patterns then (P_1 `AND` P_2), (P_1 `UNION` P_2), as well as (P_1 `MINUS` P_2), and (P_1 `OPTIONAL` P_2) are graph patterns;
– If P_1 is a graph pattern and R is a filter SPARQL expression[4] then the construction (P_1 `FILTER` R) is a graph pattern;
– If P_1 is a graph pattern and *term* is a variable or an IRI then (`GRAPH` *term* P_1) is a graph pattern.

The SPARQL 1.1 Working Draft also defines Basic Graph Patterns (BGPs), which correspond to sets of triple patterns. A Basic Graph Pattern $P_1, \ldots, P_n$ is encoded as the graph pattern (() `AND` (P_1 `AND` ($P_2 \ldots$ `AND` P_n)) ...). We ignore in this presentation the semantics of `FILTER` expressions, whose syntax is rather complex. For the purposes of this paper it is enough to consider that these expressions after evaluation return a boolean value, and therefore we also ignore errors. However, we show how to treat the `EXISTS` and `NOT EXISTS` patterns in `FILTER` expressions since these require querying graph data, and therefore provenance information should be associated to these patterns.

3.3 SPARQL Algebra

Evaluation of SPARQL patterns return multisets (bags) of solution mappings. A solution mapping, abbreviated solution, is a partial function $\mu : \mathbf{V} \to \mathbf{T}$. The

[2] Literals in the subject of triples are allowed, since this generalization is expected to be adopted in the near future.
[3] For simplicity, we do not allow blank nodes in triple patterns.
[4] For the full syntax of filter expressions, see the W3C Working Draft [1].

domain of μ is the subset of variables of $\mathbf{V}$ where μ is defined. Two mappings μ_1 and μ_2 are compatible if for every variable v in $dom(\mu_1) \cap dom(\mu_2)$ it is the case that $\mu_1(v) = \mu_2(v)$. It is important to understand that any mappings with disjoint domain are compatible, and in particular the solution mapping μ_0 with empty domain is compatible with every solution. If two solutions μ_1 and μ_2 are compatible then their union $\mu_1 \cup \mu_2$ is also a solution mapping. We represent extensionally a solution mapping as a set of pairs of the form (v, t); in the case of a solution mapping with a singleton domain we use the abbreviation $v \to t$. Additionally, if P is an arbitrary pattern we denote by $\mu(P)$ the result of substituting the variables in P defined in μ by their assigned values.

We denote that solution mapping μ satisfies the filter expression R with respect to the active graph G of dataset D by $\mu \models_{D(G)} R$. Including the parameter $D(G)$ in the evaluation of filter expressions is necessary in order to evaluate `EXISTS`(P) and `NOT EXISTS`(P) filter expressions, where P is an arbitrary graph pattern. If these constructs are removed from the language, then one only needs to consider the current solution mapping to evaluate expressions (as done in [10]).

Definition 1 (SPARQL algebra operators [1]). *Let Ω_1 and Ω_2 be multisets of solution mappings, and R a filter expression. Define:*

Join: $\Omega_1 \bowtie \Omega_2 = \{\!|\mu_1 \cup \mu_2 \mid \mu_1 \in \Omega_1$ *and* $\mu_2 \in \Omega_2$ *such that* μ_1 *and* μ_2 *are compatible* $|\!\}$
Union: $\Omega_1 \cup \Omega_2 = \{\!|\mu \mid \mu \in \Omega_1$ *or* $\mu \in \Omega_2|\!\}$
Minus: $\Omega_1 - \Omega_2 = \{\!|\mu_1 \mid \mu_1 \in \Omega_1$ *such that* $\forall_{\mu_2 \in \Omega_2}$ *either* μ_1 *and* μ_2 *are not compatible or* $dom(\mu_1) \cap dom(\mu_2) = \emptyset|\!\}$
Diff: $\Omega_1 \setminus_R^{D(G)} \Omega_2 = \{\!|\mu_1 \mid \mu_1 \in \Omega_1$ *such that* $\forall_{\mu_2 \in \Omega_2}$ *either* μ_1 *and* μ_2 *are not compatible, or* μ_1 *and* μ_2 *are compatible and* $\mu_1 \cup \mu_2 \not\models_{D(G)} R|\!\}$
LeftJoin: $\Omega_1 ⟕_R^{D(G)} \Omega_2 = (\Omega_1 \bowtie \Omega_2) \cup (\Omega_1 \setminus_R^{D(G)} \Omega_2)$

The **Diff** operator is auxiliary to the definition of **LeftJoin**. The SPARQL 1.1 Working Draft also introduces the notion of sequence to provide semantics to modifiers like `ORDER BY`. The semantics of the extra syntax is formalized by several more operators, namely aggregates and sequence modifiers (e.g. ordering), as well as property path expressions; we briefly discuss their treatment later on. Since lists can be seen as multisets with order and, without loss of generality regarding provenance information, we just consider multisets.

Definition 2 (SPARQL graph pattern evaluation). *Let $D(G)$ be an RDF dataset with active graph G, initially the default graph in $D(G)$. Let P, P_1 and P_2 be arbitrary graph patterns, and t a triple pattern. The evaluation of a graph pattern over $D(G)$, denoted by $[\![.]\!]_{D(G)}$ is defined recursively as follows:*

1. $[\![()]\!]_{D(G)} = \{\!|\mu_0|\!\}$;
2. $[\![t]\!]_{D(G)} = \{\!|\ \mu \mid dom(\mu) = var(t)$ *and* $\mu(t) \in G|\!\}$, *where* $var(t)$ *is the set of variables occurring in the triple pattern* t;
3. $[\![(P_1\ \texttt{AND}\ P_2)]\!]_{D(G)} = [\![P_1]\!]_{D(G)} \bowtie [\![P_2]\!]_{D(G)}$;
4. $[\![(P_1\ \texttt{UNION}\ P_2)]\!]_{D(G)} = [\![P_1]\!]_{D(G)} \cup [\![P_2]\!]_{D(G)}$;

5. $[\![(P_1 \texttt{ MINUS } P_2)]\!]_{D(G)} = [\![P_1]\!]_{D(G)} - [\![P_2]\!]_{D(G)}$;
6. $[\![(P_1 \texttt{ OPTIONAL } P_2)]\!]_{D(G)} = [\![P_1]\!]_{D(G)} \mathbin{⟕}^{D(G)}_{true} [\![P_2]\!]_{D(G)}$, *where P_2 is not a* FILTER *pattern;*
7. $[\![(P_1 \texttt{ OPTIONAL } (P_2 \texttt{ FILTER } R))]\!]_{D(G)} = [\![P_1]\!]_{D(G)} \mathbin{⟕}^{D(G)}_{R} [\![P_2]\!]_{D(G)}$;
8. $[\![(P_1 \texttt{ FILTER } R)]\!]_{D(G)} = \{\!|\, \mu \in [\![P_1]\!]_{D(G)} \mid \mu \models_{D(G)} R \,|\!\}$;
9. *Evaluation of* $[\![(\texttt{GRAPH } term\ P_1)]\!]_{D(G)}$ *depends on the form of term:*
   – *If term is an IRI corresponding to a graph name* $\texttt{u}_\texttt{i}$ *in* $D(G)$ *then* $[\![(\texttt{GRAPH } term\ P_1)]\!]_{D(G)} = [\![P_1]\!]_{D(G_i)}$;
   – *If term is an IRI that does not correspond to any graph in* $D(G)$ *then* $[\![(\texttt{GRAPH } term\ P_1)]\!]_{D(G)} = \{\!|\,|\!\}$;
   – *If term is a variable v then* $[\![(\texttt{GRAPH } term\ P_1)]\!]_{D(G)} =$

$$= ([\![P_1]\!]_{D(G_1)} \bowtie \{\!|v \rightarrow <\texttt{u}_\texttt{1}> |\!\}) \cup \ldots \cup ([\![P_1]\!]_{D(G_n)} \bowtie \{\!|v \rightarrow <\texttt{u}_\texttt{n}> |\!\})$$

The evaluation of EXISTS and NOT EXISTS is performed in the satisfies relation of filter expressions.

Definition 3. *Given a solution mapping μ and a graph pattern P over an RDF dataset $D(G)$ then $\mu \models_{D(G)}$* EXISTS(P) *(resp. $\mu \models_{D(G)}$* NOT EXISTS(P)*) iff $[\![\mu(P)]\!]_{D(G)}$ is a non-empty (resp. empty) multiset.*

Example 2. The SPARQL query of Example 1 corresponds to the following graph pattern :

$$\begin{aligned} Q = (\ & (?who, <\texttt{foaf : account}>, ?acc) \texttt{ OPTIONAL} \\ & (?acc, <\texttt{foaf : accountServiceHomepage}>, ?home) \\ ) & \end{aligned}$$

The evaluation of the query result with respect to the RDF dataset $D = \{G\}$, just containing the default graph G, specified in the example is:

$$\begin{aligned} & [\![Q]\!]_{D(G)} = \\ & [\![(?who, <\texttt{foaf : account}>, ?acc)]\!]_{D(G)} \mathbin{⟕}^{D(G)}_{true} \\ & [\![(?acc, <\texttt{foaf : accountServiceHomepage}>, ?home)]\!]_{D(G)} \\ & = \{\!|\{(?who, <\texttt{http : //people/david}>), (?acc, <\texttt{http : //bank}>)\}, \\ & \quad \{(?who, <\texttt{http : //people/felix}>), (?acc, <\texttt{http : //games}>)\}|\!\} \mathbin{⟕}^{D(G)}_{true} \\ & \quad \{\!|\{(?acc, <\texttt{http : //bank}>), (?home, <\texttt{http : //bank/yourmoney}>)\}\ |\!\} \\ & = \{\!|\{(?who, <\texttt{http : //people/david}>), (?acc, <\texttt{http : //bank}>), \\ & \qquad (?home, <\texttt{http : //bank/yourmoney}>)\}, \\ & \quad \{(?who, <\texttt{http : //people/felix}>), (?acc, <\texttt{http : //games}>)\} \\ & \quad |\!\} \end{aligned}$$

The evaluation of query Q returns, as expected, two solution mappings.

4 Translating SPARQL Algebra into Relational Algebra

The rationale for obtaining how-provenance for SPARQL is to represent each solution mapping as a tuple of a relational algebra query constructed from the original SPARQL graph pattern. The construction is intricate and fully specified, and is inspired from the translation of full SPARQL 1.0 queries into SQL, as detailed in [6], and into Datalog in [11]. Here, we follow a similar strategy but for simplicity of presentation we assume that a given RDF dataset $D = \{G_0, (<\mathtt{u_1}>, G_1), (<\mathtt{u_2}>, G2), \ldots, (<\mathtt{u_n}>, G_n)\}$ is represented by the two relations: `Graphs(gid,IRI)` and `Quads(gid,sub,pred,obj)`. The former stores information about the graphs in the dataset D where `gid` is a numeric graph identifier, and `IRI` an IRI reference. The relation `Quads` stores the triples of every graph in the RDF dataset. Different implementations may immediately adapt the translation provided here in this section to their own schema.

Relation `Graphs(gid,IRI)` contains a tuple $(i, <\mathtt{u_i}>)$ for each named graph $(<\mathtt{u_i}>, G_i)$, and the tuple $(0, < \ >)$ for the default graph, while relation `Quads(gid,sub,pred,obj)` stores a tuple of the form (i, s, p, o) for each triple $(s, p, o) \in G_i$[5]. With this encoding, the default graph always has identifier 0, and all the graph identifiers are consecutive integers.

It is also assumed the existence of a special value `unb`, distinct from the encoding of any RDF term, to represent that a particular variable is unbound in the solution mapping. This is required in order to be able to represent solution mappings as tuples with fixed and known arity. Moreover, we assume that the variables are totally ordered (e.g. lexicographically). The translation requires the full power of relational algebra, and notice that bag semantics is assumed (duplicates are allowed) in order to obey to the cardinality restrictions of SPARQL algebra operators [1].

Definition 4 (Translation of triple patterns). *Let $t = (s, p, o)$ be a triple pattern and G an attribute. Its translation $[(s, p, o)]^G_{\mathcal{R}}$ into relational algebra is constructed from relation* `Quads` *as follows:*

1. *Select the tuples with the conjunction obtained from the triple pattern by letting* `Quads.sub` $= s$ *(resp.* `Quads.pred` $= p$, `Quads.obj` $= o$*) if s (resp. p, o) are RDF terms; if a variable occurs more than once in t, then add an equality condition among the corresponding columns of* `Quads`*;*
2. *Rename* `Quads.gid` *as G; rename as many as* `Quads` *columns as distinct variables that exist in t, such that there is exactly one renamed column per variable;*
3. *Project in G and variables occurring in t;*

The empty graph pattern is translated as $[()]^G_{\mathcal{R}} = \Pi_G \left[\rho_{G \leftarrow \mathtt{gid}}(\mathtt{Graphs})\right]$.

Example 3. Consider the following triple patterns:

$$t_1 = (?who, <\mathtt{http://xmlns.com/foaf/0.1/account}>, ?acc)$$
$$t_2 = (?who, <\mathtt{http://xmlns.com/foaf/0.1/knows}>, ?who)$$
$$t_3 = (<\mathtt{http://cd}>, <\mathtt{http://xmlns.com/foaf/0.1/name}>, \text{"Carlos"@pt})$$

[5] For simplicity `sub`, `pred`, and `obj` are text attributes storing lexical forms of the triples' components. We assume that datatype literals have been normalized, and blank nodes are distinct in each graph. The only constraint is that different RDF terms must be represented by different strings; this can be easily guaranteed.

The corresponding translations into relational algebra are:

$$[t_1]^G_{\mathcal{R}} = \Pi_{G,acc,who}\left[\rho_{\substack{G \leftarrow \texttt{gid}\\ acc \leftarrow \texttt{obj}\\ who \leftarrow \texttt{sub}}}\left(\sigma_{\texttt{pred}=\texttt{<http://xmlns.com/foaf/0.1/account>}}(\texttt{Quads})\right)\right]$$

$$[t_2]^G_{\mathcal{R}} = \Pi_{G,who}\left[\rho_{\substack{G \leftarrow \texttt{gid}\\ who \leftarrow \texttt{sub}}}\left(\sigma_{\substack{\texttt{pred} = \texttt{<http://xmlns.com/foaf/0.1/knows>}\\ \wedge\\ \texttt{sub} = \texttt{obj}}}(\texttt{Quads})\right)\right]$$

$$[t_3]^G_{\mathcal{R}} = \Pi_{G}\left[\rho_{G \leftarrow \texttt{gid}}\left(\sigma_{\substack{\texttt{sub} = \texttt{<http://cd>} \wedge\\ \texttt{pred} = \texttt{<http://xmlns.com/foaf/0.1/name>} \wedge\\ \texttt{obj} = \text{"Carlos"@pt}}}(\texttt{Quads})\right)\right]$$

The remaining pattern that requires querying base relations is GRAPH:

Definition 5 (Translation of GRAPH pattern). *Consider the graph pattern* (GRAPH *term* P_1) *and let* G' *be a new attribute name.*

- *If term is an IRI then* $[(\texttt{GRAPH}\ term\ P_1)]^G_{\mathcal{R}}$ *is*

$$[()]^G_{\mathcal{R}} \bowtie \Pi_{var(P_1)}\left[\Pi_{G'}\left(\rho_{G' \leftarrow \texttt{gid}}\left(\sigma_{term=\texttt{IRI}}(\texttt{Graphs})\right)\right) \bowtie [P_1]^{G'}_{\mathcal{R}}\right]$$

- *If term is a variable v then* $[(\texttt{GRAPH}\ term\ P_1)]^G_{\mathcal{R}}$ *is*

$$[()]^G_{\mathcal{R}} \bowtie \Pi_{\{v\}\cup var(P_1)}\left[\rho_{G' \leftarrow \texttt{gid}, v \leftarrow \texttt{IRI}}\left(\sigma_{\texttt{gid}>0}(\texttt{Graphs})\right) \bowtie [P_1]^{G'}_{\mathcal{R}}\right]$$

Notice that the relational algebra query resulting from the translation of the pattern graph P_1 renames and hides the graph attribute. The join of the empty pattern is included in order to guarantee that each query returns the graph identifier in the first "column".

Definition 6 (Translation of the UNION pattern). *Consider the graph pattern* (P_1 UNION P_2). *The relation algebra expression* $[(P_1\ \texttt{UNION}\ P_2)]^G_{\mathcal{R}}$ *is:*

$$\begin{array}{c}\Pi_{G,var(P_1)\cup\{v\leftarrow\texttt{unb}|v\in var(P_2)\setminus var(P_1)\}}\left([P_1]^G_{\mathcal{R}}\right)\\ \bigcup\\ \Pi_{G,var(P_2)\cup\{v\leftarrow\texttt{unb}|v\in var(P_1)\setminus var(P_2)\}}\left([P_2]^G_{\mathcal{R}}\right)\end{array}$$

The union operator requires the use of an extended projection in order to make unbound variables which are present in one pattern but not in the other. The ordering of the variables in the projection must respect the total order imposed in the variables. This guarantees that the attributes are the same and by the same order in the resulting argument expressions of the union operator.

Definition 7 (Translation of the `AND` pattern). *Consider the graph pattern* $(P_1 \texttt{ AND } P_2)$ *and let* $var(P_1) \cap var(P_2) = \{v_1, \ldots, v_n\}$ *(which may be empty). The relational algebra expression* $[(P_1 \texttt{ AND } P_2)]^G_{\mathcal{R}}$ *is*

$$\Pi_{\substack{G,\\ var(P_1)-var(P_2),\\ var(P_2)-var(P_1),\\ v_1 \leftarrow first(v_1', v_1''), \ldots,\\ v_n \leftarrow first(v_n', v_n'')}} \left[\sigma_{comp} \left(\rho_{\substack{v_1' \leftarrow v_1\\ \vdots \\ v_n' \leftarrow v_n}} \left([P_1]^G_{\mathcal{R}} \right) \bowtie \rho_{\substack{v_1'' \leftarrow v_1\\ \vdots \\ v_n'' \leftarrow v_n}} \left([P_2]^G_{\mathcal{R}} \right) \right) \right]$$

where comp is a conjunction of conditions $v_i' = \texttt{unb} \vee v_i'' = \texttt{unb} \vee v_i' = v_i''$ *for each variable* $v_i (1 \leq i \leq n)$*. The function* $first$ *returns the first argument which is not* `unb`*, or* `unb` *if both arguments are* `unb`*. Note that if the set of common variables is empty then the relational algebra expression simplifies to:*

$$\Pi_{G, var(P_1) \cup var(P_2)} \left[[P_1]^G_{\mathcal{R}} \bowtie [P_2]^G_{\mathcal{R}} \right]$$

We need to rename common variables in both arguments, since an unbound variable is compatible with any bound or unbound value in order to be able to check compatibility using a selection (it is well-known that the semantics of `unb` is different from semantics of NULLs in relational algebra). The use of the $first$ function in the extended projection is used to obtain in the solution the bound value of the variable, whenever it exists. This technique is the same with that used in [6,11]. The use of the extended projection is not essential, since it can be translated into a more complex relational algebra query by using an auxiliary relation containing a tuple for each pair of compatible pairs of variables.

Definition 8 (Translation of the `MINUS` pattern). *Consider the graph pattern* $(P_1 \texttt{ MINUS } P_2)$ *and let* $var(P_1) \cap var(P_2) = \{v_1, \ldots, v_n\}$ *(which may be empty). The relational algebra expression* $[(P_1 \texttt{ MINUS } P_2)]^G_{\mathcal{R}}$ *is*

$$[P_1]^G_{\mathcal{R}} \bowtie \left[\delta \left([P_1]^G_{\mathcal{R}} \right) - \Pi_{G, var(P_1)} \left[\sigma_{comp \wedge \neg disj} \left([P_1]^G_{\mathcal{R}} \bowtie \rho_{\substack{v_1' \leftarrow v_1\\ \vdots \\ v_n' \leftarrow v_n}} \left([P_2]^G_{\mathcal{R}} \right) \right) \right] \right]$$

where comp is a conjunction of conditions $v_i = \texttt{unb} \vee v_i' = \texttt{unb} \vee v_i = v_i'$ *for each variable* $v_i (1 \leq i \leq n)$*, and disj is the conjunction of conditions* $v_i = \texttt{unb} \vee v_i' = \texttt{unb}$ *for each variable* $v_i (1 \leq i \leq n)$*. Note that if the set of common variables is empty then the above expression reduces to* $[P_1]^G_{\mathcal{R}}$ *since* $disj = true$*.*

This is the first of the non-monotonic SPARQL patterns, and deserves some extra explanation. We need to check dynamically if the domains of variables are disjoint since we do not know at translation time what are the unbound variables in the solution mappings, except when trivially the arguments of MINUS do not share any variable. The expression on the right hand side of the difference operator returns a tuple corresponding to a solution mapping μ_1 of P_1 whenever it is possible to find a solution mapping μ_2 of P_2 that it is compatible with μ_1 (condition *comp*) and the mappings do not have disjoint domains (condition $\neg disj$). By deleting these tuples (solutions) from solutions of P_1 we negate the condition, and capture the semantics of the MINUS operator. The use of the duplicate elimination δ ensures that only one tuple is obtained for each solution mapping, in order to guarantee that the cardinality of the result is as what is specified by SPARQL semantics: each tuple in $[P_1]^G_{\mathcal{R}}$ joins with at most one tuple (itself) resulting from the difference operation.

Definition 9 (Translation of FILTER pattern). *Consider the graph pattern* (P FILTER R)*, and let* [NOT] EXISTS(P_1), *…,* [NOT] EXISTS(P_m) *the* EXISTS *or* NOT EXISTS *filter expressions occurring in* R *(which might not occur). The relational algebra expression* $[(P \texttt{ FILTER } R)]^G_{\mathcal{R}}$ *is*

$$\Pi_{G,var(P)}\left[\sigma_{filter}\left([P]^G_{\mathcal{R}} \bowtie E_1 \bowtie \ldots \bowtie E_m\right)\right]$$

where $filter$ *is a condition obtained from* R *where each occurrence of* EXISTS(P_i) *(resp.* NOT EXISTS(P_i)*) is substituted by condition* $ex_i <> 0$ *(resp.* $ex_i = 0$*), where* ex_i *is a new attribute name. Expression* $E_i (1 \leq i \leq m)$ *is:*

$$\Pi_{G,var(P),ex_i\leftarrow 0}\left[\delta(P') - \Pi_{G,var(P)}\left(\sigma_{subst}\left(P' \bowtie \rho_{\substack{v'_1 \leftarrow v_1\\ \vdots \\ v'_n \leftarrow v_n}}(P'_i)\right)\right)\right]$$
$$\cup$$
$$\Pi_{G,var(P),ex_i\leftarrow 1}\left[\delta(P') - \left[\delta(P') - \Pi_{G,var(P)}\left(\sigma_{subst}\left(P' \bowtie \rho_{\substack{v'_1 \leftarrow v_1\\ \vdots \\ v'_n \leftarrow v_n}}(P'_i)\right)\right)\right]\right]$$

where $P' = [P]^G_{\mathcal{R}}$*,* $P'_i = [P_i]^G_{\mathcal{R}}$*, and* $subst$ *is the conjunction of conditions* $v_i = v'_i \vee v_i = \texttt{unb}$ *for each variable* v_i *in* $var(P) \cap var(P_i) = \{v_1, \ldots, v_n\}$*. Note that if there are no occurrences of* EXISTS *patterns, then* $[(P \texttt{ FILTER } R)]^G_{\mathcal{R}}$ *is* $\sigma_R\left([P]^G_{\mathcal{R}}\right)$.

The translation of FILTER expressions turns out to be very complex due to the EXISTS patterns. For each exists expression we need to introduce an auxiliary expression returning a unique tuple for each solution mapping of P, the top expression when the pattern P_i does not return any solution, and the bottom expression when it does. We need the double negation in order to not affect the cardinality of the results of the filter operation when pattern P returns more than one solution. Obviously, our translation depends on the capability of expressing arbitrary SPARQL conditions as relational algebra conditions; this is not immediate but assumed possible due to the translation provided in [6].

We can now conclude our translation by taking care of the `OPTIONAL` graph pattern, since it depends on the translation of filter patterns:

Definition 10 (Translation of `OPTIONAL` pattern). *Consider the graph pattern* $(P_1$ `OPTIONAL` $(P_2$ `FILTER` $R))$.
The relational algebra expression $[(P_1 \texttt{ OPTIONAL } (P_2 \texttt{ FILTER } R))]_{\mathcal{R}}^{G}$ *is*

$$\begin{gathered}[(P_1 \texttt{ AND } P_2)]_{\mathcal{R}}^{G} \\ \bigcup \\ \Pi_{G,var(P_1)\cup\{v\leftarrow\texttt{unb}|v\in var(P_2)\setminus var(P_1)\}} \\ \left[[P_1]_{\mathcal{R}}^{G} \bowtie \left(\begin{array}{c} \delta\left([P_1]_{\mathcal{R}}^{G}\right) \\ - \\ \Pi_{G,var(P_1)}\left([(P_1 \texttt{ AND } P_2) \texttt{ FILTER } R]_{\mathcal{R}}^{G}\right) \end{array} \right) \right]\end{gathered}$$

The translation of $(P_1$ `OPTIONAL` $P_2)$ *is obtained from the translation of the graph pattern* $(P_1$ `OPTIONAL` $(P_2$ `FILTER` *true*$))$.

The translation of the `OPTIONAL` pattern has two parts, one corresponding to the `JOIN` operator (top expression) and one corresponding to the **Diff** operator. The translation of the **Diff** operator uses the same technique as the `MINUS` operator but now we remove from solutions of P_1 those solution mappings of P_1 that are compatible with a mapping of P_2 and that satisfy the filter expression.

Theorem 1 (Correctness of translation). *Given a graph pattern P and a RDF dataset $D(G)$ the process of evaluating the query is performed as follows:*

1. *Construct the base relations* `Graphs` *and* `Quads` *from* $D(G)$*;*
2. *Evaluate* $[SPARQL(P, D(G), V)]_{\mathcal{R}} = \Pi_V \left[\sigma_{G'=0}\left([()]_{\mathcal{R}}^{G'} \bowtie [P]_{\mathcal{R}}^{G'}\right)\right]$ *with respect to the base relations* `Graphs` *and* `Quads`*, where G' is a new attribute name and $V \subseteq var(P)$.*

Moreover, the tuples of relational algebra query (2) are in one-to-one correspondence with the solution mappings of $[\![P]\!]_{D(G)}$ when $V = var(P)$, and where an attribute mapped to `unb` *represents that the corresponding variable does not belong to the domain of the solution mapping.*

Proof. The proof is by by structural induction on the graph patterns and can be found in the extended version of this paper available at `http://arxiv.org/abs/1209.0378`.

The constructed translation will be used to extract how-provenance information for SPARQL queries, addressing the problems identified in [12].

5 Provenance for SPARQL Queries

The crux of the method has been specified in the previous section, and relies on the properties of the extended provenance m-semiring $\mathcal{K}_{dprovd}$ for language $\mathcal{RA}^{+}_{\mathcal{K}}(-,\delta)$. We just need a definition before we illustrate the approach.

Definition 11 (Provenance for SPARQL). *Given a graph pattern P and a RDF dataset $D(G)$ the provenance for P is obtained as follows:*

- *Construct the base $\mathcal{K}_{dprovd}$-relations by annotating each tuple in* `Graphs` *and* `Quads` *with a new identifier;*
- *Construct an annotated query $SPARQL(P, D(G), V)_{\mathcal{K}_{dprovd}}$ from relational algebra $[SPARQL(P, D(G), V)]_{\mathcal{R}}$ expression by substituting the duplicate elimination operator by δ_1 where 1 is the identity element of $\mathcal{K}_{dprovd}$.*

The provenance information for P is the annotated relation obtained from evaluating $SPARQL(P, D(G), V)_{\mathcal{K}_{dprovd}}$ with respect to the annotated translation of the dataset $D(G)$.

By the factorization property of $\mathcal{K}_{dprovd}$ we know that this is the most general m-semiring, and thus the provenance obtained according to Definition 11 is the most informative one. We just need to illustrate the approach with Example 1 in order to completely justify its appropriateness.

Example 4. First, we represent the RDF dataset by $\mathcal{K}_{dprovd}$-relations where the annotation tags are shown in the last column. The IRIs have been abbreviated:

`Graphs`

gid	IRI	
0	< >	g_0

`Quads`

gid	sub	pred	obj	
0	<david>	<account>	<bank>	t_1
0	<felix>	<account>	<games>	t_2
0	<bank>	<accountServiceHomepage>	<bank/yourmoney>	t_3

Returning to query $Q = (Q_1$ `OPTIONAL` $Q_2)$ of Example 2 with (sub)patterns $Q_1 = (?w, <\texttt{account}>, ?a)$ and $Q_2 = (?a, <\texttt{accountServiceHomepage}>, ?h)$, we obtain the following expressions for Q_1 and Q_2:

$$[Q_1]^G_{\mathcal{R}} = \Pi_{G,w,a}\left[\rho_{\substack{G \leftarrow \texttt{gid}\\ w \leftarrow \texttt{sub}\\ a \leftarrow \texttt{obj}}}\left(\sigma_{\texttt{pred}=<\texttt{account}>}(\texttt{Quads})\right)\right]$$

$$[Q_2]^G_{\mathcal{R}} = \Pi_{G,a,h}\left[\rho_{\substack{G \leftarrow \texttt{gid}\\ a \leftarrow \texttt{sub}\\ h \leftarrow \texttt{obj}}}\left(\sigma_{\texttt{pred}=<\texttt{accountServiceHomepage}>}(\texttt{Quads})\right)\right]$$

returning the annotated relations:

$$[Q_1]^G_{\mathcal{R}} = \begin{array}{lll||l} G & w & a & \\ \hline 0 & <\texttt{david}> & <\texttt{bank}> & t_1 \\ 0 & <\texttt{felix}> & <\texttt{games}> & t_2 \end{array}$$

$$[Q_2]^G_{\mathcal{R}} = \begin{array}{lll||l} G & a & h & \\ \hline 0 & <\texttt{bank}> & <\texttt{bank/yourmoney}> & t_3 \end{array}$$

The expression $[(Q_1 \texttt{ AND } Q_2)]^G_{\mathcal{R}}$ used in the construction of the expression for the `OPTIONAL` pattern is:

$$\Pi_{G,w,a \leftarrow first(a',a''),h} \left[\sigma_{a'=a'' \vee a'=\texttt{unb} \vee a''=\texttt{unb}} \left(\rho_{a'' \leftarrow a} \left([Q_1]^G_{\mathcal{R}} \right) \bowtie \rho_{a' \leftarrow a} \left([Q_2]^G_{\mathcal{R}} \right) \right) \right]$$

obtaining the annotated relation:

$$[(Q_1 \texttt{ AND } Q_2)]^G_{\mathcal{R}} = \begin{array}{llll||l} G & w & a & h & \\ \hline 0 & <\texttt{david}> & <\texttt{bank}> & <\texttt{bank/yourmoney}> & t_1 \times t_3 \end{array}$$

We also need to determine the value of $\delta_1([Q_1]^G_{\mathcal{R}})$ which is simply:

$$\delta_1([Q_1)]^G_{\mathcal{R}}) = \begin{array}{lll||l} G & w & a & \\ \hline 0 & <\texttt{david}> & <\texttt{bank}> & 1 \\ 0 & <\texttt{felix}> & <\texttt{games}> & 1 \end{array}$$

We can now construct the expression corresponding to the **Diff** operator of SPARQL algebra, namely:

$$\Pi_{G,w,a,h \leftarrow \texttt{unb}} \left[[Q_1]^G_{\mathcal{R}} \bowtie \left(\begin{array}{c} \delta_1 \left([Q_1]^G_{\mathcal{R}} \right) \\ - \\ \Pi_{G,w,a} \left([(Q_1 \texttt{ AND } Q_2)]^G_{\mathcal{R}} \right) \end{array} \right) \right]$$

returning the annotated tuples:

G	w	a	h	
0	<`david`>	<`bank`>	`unb`	$t_1 \times (1 - (t_1 \times t_3))$
0	<`felix`>	<`games`>	`unb`	$t_2 \times (1 - 0) = t_2$

This is the important step, since K-relations assign an annotation to every possible tuple in the domain. If it is not in the support of the relation, then it is tagged with 0. Therefore, the solutions for $([(Q_1 \texttt{ AND } Q_2)]^G_{\mathcal{R}}$ are:

G	w	a	h	
0	<`david`>	<`bank`>	<`bank/yourmoney`>	$t_1 \times t_3$
0	<`david`>	<`bank`>	`unb`	$t_1 \times (1 - (t_1 \times t_3))$
0	<`felix`>	<`games`>	`unb`	t_2

and for our query, finally we get

w	a	h	
<`david`>	<`bank`>	<`bank/yourmoney`>	$g_0 \times t_1 \times t_3$
<`david`>	<`bank`>	`unb`	$g_0 \times t_1 \times (1 - (t_1 \times t_3))$
<`felix`>	<`games`>	`unb`	$g_0 \times t_2$

The interpretation of the results is the expected and intuitive one. Suppose that (i) we use the boolean m-semiring, with just the two values `t` and `f`, meaning that we trust or not trust a triple, (ii) product corresponds to conjunction, (iii) sum corresponds to disjunction, and (iv) difference is defined as $x - y = x \wedge \neg y$. So, if we trust g_0 and t_1, t_2 and t_3 we are able to conclude that we trust the first and third solutions (substitute 1 and the identifiers of trusted triples by `t` in the annotations, and then evaluate the resulting boolean expression). If we do not trust t_3 but trust the other triples then we trust the second and third solutions. Also mark how the graph provenance is also annotated in our solutions. Accordingly, if we don't trust the default graph then we will not trust any of the solutions. Therefore, our method was capable of keeping in the same annotated $\mathcal{K}_{dprovd}$-relation the several possible alternative solutions, one in each distinct tuple. This was claimed to not be possible in [12].

6 Discussion and Conclusions

The literature describes several approaches to extract data provenance/annotated information from RDF(S) data [7,5,4,12,3]. A first major distinction is that we extract how-provenance instead of only why-provenance[6] of [7,5,4,3]. Both [7,4] address the problem of extracting data provenance for RDF(S) entailed triples, but do not support SPARQL. The theory developed in [5] implements the difference operator using a negation, but it does not handle duplicate solutions according to the semantics of SPARQL because of idempotence of sum; additionally, the proposed difference operator to handle why-provenance discards the information in the right hand argument. The most complete work is [3] which develops a framework for annotated Semantic Web data, supporting RDFS entailment and providing a query language extending many of the SPARQL features in order to deal with annotated data, exposing annotations at query level via annotation variables, and including aggregates and subqueries (but not property path patterns). However, the sum operator is idempotent in order to support RDFS entailment, and by design the `UNION` operator is not interpreted in the annotation domain. Moreover, the `OPTIONAL` graph pattern discards in some situations the information in the second argument, and thus cannot extract full provenance information.

The capability of extracting full data how-provenance for SPARQL semantics as prescribed in [12] has been shown possible with our work, refuting their claim that existing algebras could not be used for SPARQL. Our approach, like [12], rests on a translation of SPARQL into annotated relational algebra contrasting with the abstract approach of [7,5,4,3]. The authors in [12] argue that this translation process does not affect the output provenance information for the case of (positive) SPARQL. In this way, the major constructs of SPARQL 1.1 are taken care respecting their bag semantics. However, contrary to the works of [7,3] we do not address the RDF schema entailment rules, and therefore our work is only applicable to simple entailment.

[6] We use the terminology "how-" and "why-provenance" in the sense of [9].

We plan to address the complete semantics of SPARQL. In particular, aggregates can be handled by summing ($\oplus$) tuples for each group, while property path patterns can generate annotation corresponding to products ($\otimes$) of the involved triples in each solution. This extension is enough to be able to capture data provenance for RDFS entailment. We also want to explore additional applications in order to assess fully the potential of the proposed method.

References

1. SPARQL 1.1 query language, 2012. W3C Working Draft (January 05, 2012), http://www.w3.org/TR/2012/WD-sparql11-query-20120105/
2. Amer, K.: Equationally complete classes of commutative monoids with monus. Algebra Universalis 18, 129–131 (1984)
3. Antoine Zimmermann, A.P., Lopes, N., Straccia, U.: A general framework for representing, reasoning and querying with annotated semantic web data. Journal of Web Semantics 11, 72–95 (2012)
4. Buneman, P., Kostylev, E.V.: Annotation algebras for RDFS. In: Proc. of the 2nd Int. Ws. on the Role of Semantic Web in Provenance Management (SWPM 2010). CEUR Workshop Proceedings (2010)
5. Dividino, R., Sizov, S., Staab, S., Schueler, B.: Querying for provenance, trust, uncertainty and other meta knowledge in RDF. Web Semant. 7(3), 204–219 (2009)
6. Elliott, B., Cheng, E., Thomas-Ogbuji, C., Ozsoyoglu, Z.M.: A complete translation from SPARQL into efficient SQL. In: Proc. of the 2009 Int. Database Engineering & Applications Symposium, IDEAS 2009, pp. 31–42. ACM (2009)
7. Flouris, G., Fundulaki, I., Pediaditis, P., Theoharis, Y., Christophides, V.: Coloring RDF Triples to Capture Provenance. In: Bernstein, A., Karger, D.R., Heath, T., Feigenbaum, L., Maynard, D., Motta, E., Thirunarayan, K. (eds.) ISWC 2009. LNCS, vol. 5823, pp. 196–212. Springer, Heidelberg (2009)
8. Geerts, F., Poggi, A.: On database query languages for K-relations. J. Applied Logic 8(2), 173–185 (2010)
9. Green, T.J., Karvounarakis, G., Tannen, V.: Provenance semirings. In: Proc. of PODS 2007, pp. 31–40. ACM, New York (2007)
10. Pérez, J., Arenas, M., Gutierrez, C.: Semantics and complexity of SPARQL. ACM Trans. Database Syst. 34(3), 16:1–16:45 (2009)
11. Polleres, A.: From SPARQL to rules (and back). In: Williamson, C.L., Zurko, M.E., Patel-Schneider, P.F., Shenoy, P.J. (eds.) Proc. of the 16th Int. Conf. on World Wide Web, WWW 2007, pp. 787–796. ACM (2007)
12. Theoharis, Y., Fundulaki, I., Karvounarakis, G., Christophides, V.: On provenance of queries on semantic web data. IEEE Internet Computing 15(1), 31–39 (2011)

SRBench: A Streaming RDF/SPARQL Benchmark

Ying Zhang[1], Pham Minh Duc[1], Oscar Corcho[2], and Jean-Paul Calbimonte[2]

[1] Centrum Wiskunde & Informatica, Amsterdam, The Netherlands
{Y.Zhang,P.Minh.Duc}@cwi.nl
[2] Universidad Politécnica de Madrid, Spain
ocorcho@fi.upm.es, jp.calbimonte@upm.es

Abstract. We introduce *SRBench*, a general-purpose benchmark primarily designed for streaming RDF/SPARQL engines, completely based on real-world data sets from the Linked Open Data cloud. With the increasing problem of too much streaming data but not enough tools to gain knowledge from them, researchers have set out for solutions in which Semantic Web technologies are adapted and extended for publishing, sharing, analysing and understanding streaming data. To help researchers and users comparing streaming RDF/SPARQL (strRS) engines in a standardised application scenario, we have designed SRBench, with which one can assess the abilities of a strRS engine to cope with a broad range of use cases typically encountered in real-world scenarios. The data sets used in the benchmark have been carefully chosen, such that they represent a realistic and relevant usage of streaming data. The benchmark defines a concise, yet comprehensive set of queries that cover the major aspects of strRS processing. Finally, our work is complemented with a functional evaluation on three representative strRS engines: SPARQL$_{\text{Stream}}$, C-SPARQL and CQELS. The presented results are meant to give a first baseline and illustrate the state-of-the-art.

1 Introduction

Unlike the static data, which are known *a priori* and rarely change, streaming data arrive as continuous streams typically at high rates, e.g., once per second or even higher. For data streams, the most recent data are usually most relevant, and the queries mainly focus on the continuous changes of the observed properties over time. The amount of streaming data has been growing extremely fast in the past years and is expected to grow even faster in the coming decades. However, existing Data Stream Management Systems (DSMSs) are not able to capture *all* information from the available streaming data, letting alone interlinking those data with other data sets to derive implicit information [5,16]. In the meantime, Semantic Web techniques have focused on how to publish and interlink data on the World Wide Web, and how to perform complex reasoning tasks on the data. However, these techniques have generally not taken into account rapidly changing streaming data. The lack of integration and communication between different streaming data resources often isolates important data streams and intensifies the existing problem of "too much (streaming) data but not enough (tools to gain and derive) knowledge" [32]. To tackle this problem, researchers have set out for solutions in which Semantic Web techniques are adapted and extended for publishing, sharing, analysing and understanding of streaming data.

Sheth et al. [32] first envisioned a Semantic Sensor Web (SSW), in which sensor data are annotated with semantic metadata to increase interoperability and provide contextual information essential for situational knowledge[1]. Subsequently, Corcho et al. [14] identified the five most relevant challenges of the current SSW. Della Valle et al. [16] proposed a novel approach, called *stream reasoning*, to provide the abstractions, foundations, methods and tools required to integrate data streams, the Semantic Web and reasoning systems. Sequeda et al. [31] introduced the concept of Linked Stream Data (LSD) which applies the Linked Data principles to streaming data, so that data streams can be published as part of the Web of Linked Data. So far, these visions have been answered by various proposals to address the topic of *streaming data processing using Semantic Web technologies* from different angles. For instance, how to apply reasoning on streaming data [1,16,32,35,36]; how to publish raw streaming data and connect them to the existing data sets on the Semantic Web [10,14,26,31,32]; and how to extend the SPARQL query language to process streaming data [6,9,11,20,24,25]. The increasing interest in streaming RDF/SPARQL (strRS) engines calls for a *standard way* to compare the functionality and performance of different systems.

So far, little work has been done on benchmarking DSMSs. The Linear Road benchmark [3] is the only publicly available DSMSs benchmark. However, it is not ideal when used to assess strRS engines. As originally designed to evaluate traditional DSMSs, the benchmark is based on the relational data model, so it does not capture the properties of RDF graph data. Moreover, Linear Road does not consider interlinking the benchmark data set with other data sets; neither does it address reasoning. In Semantic Web, existing RDF/SPARQL benchmarks, e.g., [8,21,30], have been focused on static data, so they do not capture the aforementioned dynamic properties of streaming data. In [24,25], some microbenchmark queries are used for preliminary evaluations of the proposed strRS systems. However, the queries were created with a particular system in mind and they only cover a small subset of the features of SPARQL. Hence, they cannot serve as general-purpose benchmarks.

In this paper, we present ***SRBench***, a streaming RDF/SPARQL benchmark that aims at assessing the abilities of strRS engines in dealing with important features from both DSMSs and Semantic Web research areas combined in one real-world application scenario. That is, how well can a system cope with a broad range of different query types in which Semantic Web technologies, including querying, interlinking, sharing and reasoning, are applied on highly dynamic streaming RDF data. The benchmark can help both researchers and users to compare strRS engines in a pervasive application scenario in our daily life, i.e., querying and deriving information from weather stations. To the best of our knowledge, SRBench is the first *general-purpose* benchmark that is *primarily* designed to compare strRS engines.

Given the importance of interlinked data sets in Semantic Web, and the study of Duan et al. [17], which points out that the synthetic data used by the existing RDF benchmarks generally do not accurately predict the behaviour of RDF stores in realistic scenarios, we decided to use a real-world sensor data set, i.e., LinkedSensorData [27], from the Linked Open Data (LOD) cloud [34] as the basic data set of SRBench. To

[1] *Situational knowledge* is the knowledge specific to a particular situation.

assess a system's ability of dealing with interlinked data, we additionally use the LOD data sets GeoNames [18] and DBpedia [15], which are linked to the LinkedSensorData.

SRBench defines a concise, yet comprehensive set of queries which covers the major aspects of strRS query processing, ranging from simple graph pattern matching queries only on streaming data to queries requiring reasoning over multiple interlinked data sets. Each query is intended to challenge a particular aspect of the query processor. The main advantages of applying Semantic Web technologies on streaming data include providing better search facilities by adding semantics to the data, reasoning through ontologies, and integration with other data sets. The ability of a strRS engine to process these distinctive features is accessed by the benchmark with queries that apply reasoning not only over the streaming sensor data, but also over the sensor metadata and the two aforementioned LOD data sets.

Given that existing strRS engines are still in their infancy, we deem it important to first conduct a functional evaluation. Do they provide a sufficient set of functions that are required by the streaming applications? Do they miss any crucial functionalities? Do they provide any additional functionalities that can be beneficial for streaming applications, which thus distinguish themselves from similar systems? Therefore, we complement our work on SRBench by a functional evaluation on three strRS engines, $\text{SPARQL}_{\text{Stream}}$ [11], C-SPARQL [6] and CQELS [25]. Each of these systems also proposes its own SPARQL extension for streaming data processing. The evaluation is not meant to be an exhaustive examination of all existing strRS systems. The testing systems are chosen, because they represent different approaches in strRS processing. $\text{SPARQL}_{\text{Stream}}$ aims at enabling ontology-based access to streaming data. C-SPARQL attempts to facilitate reasoning upon rapidly changing information. CQELS is the only native strRS system built from scratch. The evaluation results are intended to give a first baseline and illustrate the state-of-the-art.

The target audience of this paper can be divided into three groups. First, the framework presented here can help strRS engine implementers to verify and refine their query processors by comparing them to other implementations. Second, users can be assisted in choosing between products by using SRBench as a simple case study or pilot project that yet provides essential ingredients of the targeted system. For researchers, lastly, we provide a framework for helping to tailor existing technologies for use in streaming settings and for refinement or design of algorithms.

This paper is further organised as follows. Section 2 discusses design challenges of the benchmark. Section 3 describes the benchmark data sets. Section 4 defines the benchmark queries. Section 5 presents the results of the functional evaluation. Finally, we discuss related work in Section 6 and conclude in Section 7.

2 Design Challenges

In this section, we discuss the unique challenges that streaming RDF/SPARQL processing imposes on the design of a benchmark and how they are met by SRBench.

Proper Benchmark Data Set. First of all, the design of a streaming RDF/SPARQL benchmark requires a cautiously chosen data set that is relevant [19], realistic [17], semantically valid [3] and interlinkable [31]. Additionally, the data set should allow the formulation of queries that both feel natural and present a concise but complete set of challenges that strRS engines should meet.

In SRBench, this challenge is met by choosing the LinkedSensorData [27] data set from the LOD cloud as the basic data set. Among different kinds of streaming data[2], sensor data is a major class of streaming data with the longest history. Weather information applications have long become pervasive in our daily life, in addition, they are gaining increasing social and financial values in more accurate prediction of extreme weather conditions. The LinkedSensorData is a real-world data set containing the US weather data published by Kno.e.sis[3] according to the LSD principles [13], which applies the well-established Linked Data principles [7] to streaming data. The LinkedSensorData is the first and, so far, largest LSD data set in the LOD cloud [34] and CKAN [12] containing ~1.7 billion triples.

To assess a system's ability of dealing with interlinked data, we additionally use other data sets from the LOD cloud. Currently, this includes the GeoNames [18] and DBpedia [15] data sets. Our choice for the GeoNames data set is determined by the fact that the LinkedSensorData data set links the sensor locations to nearby geographic places defined by the GeoNames data set. The choice for the DBpedia data set is a matter of course, since DBpedia is the largest and most popularly used data set in the LOD cloud. By using the LOD data sets, it is easy to extend the benchmark with more data sets in the future. This enables adding more semantics to the benchmark's application scenario, which subsequently allows more use cases.

A Concise Set of Features. The main advantages of applying Semantic Web technologies on streaming data include providing better search and sharing facilities by adding semantics to the data, reasoning through ontologies, and integration with other data sets. The benchmark should provide a comprehensive set of queries that assess a system's ability of processing these distinctive features on highly dynamic (in terms of arriving rate and amount) streaming data, possibly in combination with static data. The queries should have different levels of complexity, so that the benchmark can be used to evaluate not only general purpose systems supporting a broad spectrum of features, but also specialised systems aiming at providing a limited number of features with high efficiency. Nonetheless, as stated by the "20 queries" principles [23], the number of queries should be compact. Thus, the number and types of queries should exhibit a good balance between conciseness and detail making it possible to run the benchmark in an acceptable time, while still acquiring interesting characteristics of the system(s) tested.

In SRBench, this challenge is met by a set of seventeen queries that have been carefully chosen such that they provide valuable insights that can be generally applied to strRS systems and are useful in many domains, e.g., notion of time bounded queries (e.g., data in the latest *X* units-of-time); notion of continuous queries (i.e., queries evaluated periodically); data summarisation in the queries (e.g., aggregates); providing high-level information from raw-data (e.g., ask for hurricanes, while the raw-data are simply temperature, wind measurements); and combining streams with contextual static data. Before designing the queries, we first identified the set of important features in the SPARQL 1.1 language [22] and streaming data processing. Then, use cases are

[2] Next to sensor data streams, there are text streams and video streams.
[3] http://knoesis.wright.edu

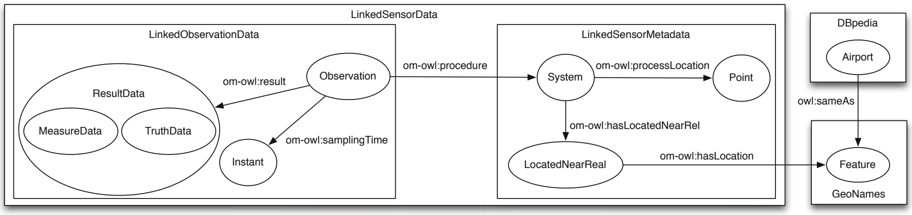


Fig. 1. An overview of the data sets used in SRBench and their relationships

carefully chosen such that they reflect how the weather information is used in the real world, while each of them challenges the query processor, with focus on one or two of the important features (but not limited to).

No Standard Query Language. A standard query language for streaming data processing has never come into existence. Therefore, the queries of a streaming benchmark should be specified in a language agnostic way, yet have a clear semantics.

In SRBench, this challenge is met by first giving a descriptive definition of the benchmark queries, in a similar way as how the Berlin SPARQL Benchmark describes its queries[4]. Then, we provide implementations of the benchmark queries using the three major SPARQL extensions for streaming data processing (for short: *streaming SPARQL*), i.e., SPARQL$_{\text{Stream}}$, C-SPARQL and CQELS. Thus, these three sets of implementing queries are not only used by the functional evaluation in Section 5, but also for the purpose of clarifying the benchmark query definitions.

3 Data Sets

In this section, we briefly describe the three LOD data sets used by SRBench. An overview of the data sets and their ontologies, and how they are linked to each other is shown in Figure 1. More information of the data sets can be found in [37].

The LinkedSensorData Data Set. Work on producing Linked Data from data emitted by sensors was initiated in 2009, pioneered by [31,26]. The LinkedSensorData contains the US weather data collected since 2002 by MesoWest[5], and were transformed into LSD by Kno.e.sis. LinkedSensorData contains two sub-datasets. The *LinkedSensorMetadata* contains expressive descriptions of ~20,000 weather stations in the US. On average, there are five sensors per weather station, so there are in total ~100,000 sensors in the data set. The sensors measure phenomena such as temperature, visibility, precipitation, pressure, wind speed and humidity. In addition to location attributes, e.g., latitude, longitude, and elevation, there are also links to locations in GeoNames that are near the weather stations. The *LinkedObservationData* contains expressive descriptions of hurricane and blizzard observations in the US. The observations collected include values of all phenomena measured by the sensors. The data set includes observations within the entire US during the time periods that several major storms were

[4] http://www4.wiwiss.fu-berlin.de/bizer/BerlinSPARQLBenchmark/
[5] http://mesowest.utah.edu/

Table 1. Statistics of the LinkedObservationData data sets used by SRBench

Name	Storm Type	Date	#Triples	#Observations	Data size
ALL			1,730,284,735	159,460,500	~111 GB
Bill	Hurricane	Aug. 17 – 22, 2009	231,021,108	21,272,790	~15 GB
Ike	Hurricane	Sep. 01 – 13, 2008	374,094,660	34,430,964	~34 GB
Gustav	Hurricane	Aug. 25 – 31, 2008	258,378,511	23,792,818	~17 GB
Bertha	Hurricane	Jul. 06 – 17, 2008	278,235,734	25,762,568	~13 GB
Wilma	Hurricane	Oct. 17 – 23, 2005	171,854,686	15,797,852	~10 GB
Katrina	Hurricane	Aug. 23 – 30, 2005	203,386,049	18,832,041	~12 GB
Charley	Hurricane	Aug. 09 – 15, 2004	101,956,760	9,333,676	~7 GB
	Blizzard	Apr. 01 – 06, 2003	111,357,227	10,237,791	~2 GB

active, including Hurricane Katrina, Ike, Bill, Bertha, Wilma, Charley, Gustav, and a major blizzard in Nevada in 2003. These observations are generated by weather stations described in the LinkedSensorMetadata data set introduced above. Currently, this data set contains almost two billion RDF triples, which together describe more than 159 million observations. For SRBench, we have obtained all linked sensor observation data sets from the original Kno.e.sis site for LinkedSensorData [27]. Table 1 shows the statistics of the LinkedObservationData data sets as presented on the original website, to which we have added the sizes of the data sets after they have been unpacked.

All data are described according to the *sensor-observation ontology* [27]. The ontology class `System` describes a weather sensor station, e.g., its ID and location of the station, a geographical location to which the station is located nearby and the weather properties observed by this station. The class `Observation` describes an observation made by a weather sensor station, e.g., the ID of the weather station that has made the observation, the type of the observed weather property and the value and time of the observation. The class `MeasureData` describes the numerical value of an observation, while the class `TruthData` describes the truth-value of an observation. The class `LocatedNearRel` describes a geographic location to which a weather sensor station is located nearby, e.g., the distance between the nearby location and the sensor station. The class `Point` describes a geographic point location, in terms of latitude, longitude and altitude. The class `Instant` describes a date/time object.

The GeoNames Data Set is a free geographical database that covers all countries and contains >8 million place names [18]. For SRBench, we use version 3.01 of the *GeoNames ontology* [18]. Its main class is `Feature`, which describes a geographical location, e.g., its names, its latitude and longitude, the country to which this location belong and other locations that are close to this location. We have obtained the dump of the complete GeoNames RDF data set, which contains ~8 million geographic features with ~46 million RDF triples. The dump has one RDF document per toponym. The complete data set occupies ~10GB on disk.

The DBpedia Data Set. The DBpedia ontology is highly complex but well documented, so we do not repeat its class definitions here, but refer the interested readers to its official website [15] instead. For SRBench, we have obtained the data sets from the English language collection, which consists of 44 RDF files in N-triple format with ~181 million triples. The total data size is ~27 Gigabytes. The DBpedia data set is di-

Table 2. Addressed features per query. Operators are abbreviated in per row unique capital letters, defined as: 1. **A**nd, **F**ilter, **U**nion, **O**ptional; 2. **P**rojection, **D**istinct; 3. **S**elect, **C**onstruct, **A**sk; 4. **A**ggregate, **S**ubquery, **N**egation, **E**xpr in `SELECT`, assign**M**ent, **F**unctions&operators, **P**roperty path; 5. sub**C**lassOf, subp**R**opertyOf, owl:same**A**s; 6. **T**ime-based window, **I**stream, **D**stream, **R**stream; 7. Linked**O**bservationData, Linked**S**ensorMetadata, **G**eoNames, **D**bpedia.

	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q12	Q13	Q14	Q15	Q16	Q17
1 graph pattern matching	A	A,F,O	A	A,F	A	A,F,U	A	A	A	A	A,F	A,F,U	A,F	A,F,U	A,F	A,F	A,F
2 solution modifier	P,D	P,D	P	P	P	P	P,D	P	P	P,D	P,D	P	P	P,D	P	P	P
3 query form	S	S	A	S	C	S	S	S	S	S	S	S	S	S	S	S	S
4 SPARQL 1.1		F,P	A	A,E,M,F	A,S		N	A,E,M	A,E,M		A,S,M,F	A,S,E,M,F,P	A,E,M,F,P	F,P	A,E,M,P	P	P
5 reasoning			R												C	A	C
6 streaming feature	T	T	T	T	T	T	T,	T	T	T	T	T	T	T	T		
7 data access	O	O	O	O	O	O	O	O, S	O,S	O,S	O,S	O,S,F	O,S,G	O,S,G	O,S,D	O,S,G,D	S

rectly linked to the GeoNames data set through the `owl:sameAs` property. DBpedia has in total 85,000 links to the GeoNames data set.

4 Benchmark Queries

In this section, we define the SRBench benchmark queries. An overview of the language features addressed by each query is given in Table 2.

A SPARQL query can be divided into three parts. As SPARQL is essentially a graph-matching query language, *graph pattern matching* is the fundamental and one of the most complex parts of a SPARQL query. This part includes features such as the basic graph pattern matching operators '.' (representing a natural join `AND`) and `FILTER`, and the most complicated operators `UNION` and `OPTIONAL` [28]. All graph pattern matching operators are addressed in the SRBench queries.

The second part is called the *solution modifiers*, which are used to modify the results of the graph pattern matching operators. The solution modifiers contain six operators, i.e., projection, `DISTINCT`, `ORDER BY`, `OFFSET`, `LIMIT` and `REDUCED`. In the SRBench queries, only the projection and `DISTINCT` solution modifiers are addressed, because the additional values of the other four operators are negligible in streaming applications. `ORDER BY` is ignored since streaming data are already sorted by their time stamps, and sorting the results on another attribute will only produce partially sorted data (within one window). The features of `OFFSET` and `LIMIT` are largely covered by sliding windows, which are more appropriate for strRS queries. Finally, the nondeterministic property of `REDUCED` highly complicates the verification of the query results.

The last part is called the *query forms*, which determine the form of the final output of a SPARQL query. The output can be one of the four query forms: `SELECT` returns the projected variables bound in a query pattern match; `CONSTRUCT` returns a new RDF graph constructed by substituting variables in a set of triple templates; `ASK` returns a boolean indicating whether a query pattern matches or not; and `DESCRIBE` returns an RDF graph that describes the resources found. In the SRBench queries, the `DESCRIBE` form is not used, because the result of a `DESCRIBE` query is highly implementation dependant, which largely complicates the verification of the query results. Moreover, the functionality of `DESCRIBE` can be approximated using explicit graph pattern matching and projections. The first three rows of Table 2 respectively survey how the operators of the three parts are distributed among the benchmark queries.

Next to the features defined by SPARQL 1.0 [29], SPARQL 1.1 has introduced several new features, including aggregates, subqueries, negation, expressions in the `SELECT` clause, Property Paths, assignment, a short form for `CONSTRUCT`, and an expanded set of functions and operators. Except the short form of `CONSTRUCT`, which is merely a syntax sugar, we make extensive use of these new features in the benchmark queries, especially the Property Paths expressions, which we regard as a major contribution of SPARQL 1.1 that provides great flexibility to navigate through the RDF graphs. Note that, since SPARQL 1.1 is still a W3C working draft, changes to the syntax and/or semantics of the new language features are possible. For instance, the semantics of the Property Path expressions might be modified, due to recent analysis of the unfeasibility of their current semantics [4]. Possible changes in SPARQL 1.1 will not affect the definition of the benchmark queries, since they are specified independent of any query language. Row 4 of Table 2 surveys how the SPARQL 1.1 new features are distributed among the benchmark queries.

A main added value of applying Semantic Web technologies on streaming data is the possibility of reasoning over the data, so SRBench includes queries that allow exploiting such facility if provided by the processing engine. Currently, the queries involve reasoning over the `rdfs:subClassOf`, `rdfs:subPropertyOf` and `owl:sameAs` properties. Note that, these queries can be implemented and executed by both systems with and without inference mechanisms, but the differences might be noticeable in the query results. That is, systems with inference mechanisms will most probably return more results than systems without such mechanisms. Also note that, although SPARQL is not a reasoning language, it can be used to query ontologies if they are encoded in RDF. So, on systems without reasoning, this shortcoming can be alleviated by explicitly expressing reasoning tasks using extra graph patterns with Property Path over the ontologies. In our functional evaluation (Section 5), we actually use this workaround to implement the benchmark queries using the three language extensions tested. Row 5 of Table 2 surveys how reasoning features are distributed among the benchmark queries.

Although there is no standard query language for streaming RDF data, existing streaming SPARQL extensions generally introduce streaming data operators that are inspired by the continuous query language CQL [2]. So, next to the classical SPARQL 1.0 and 1.1 operators, we add three important streaming SPARQL features. The *time-based sliding window* operator is a basic operator in streaming data processing, which allows users to control data access using time intervals. A slide size can be defined to create overlapping or disjoint windows. The *window-to-stream* operators, *Istream* and *Dstream*, return data items that have been inserted or deleted since the previous window, respectively. Although not frequently used, these operators help to detect changes in the data streams, a feature particularly important for streaming data. Row 6 of Table 2 surveys how the streaming operators are distributed among the benchmark queries.

To conclude, the SRBench queries are designed based on a real use case in LSD. They cover the most important SPARQL operators and the common streaming SPARQL extensions. SRBench clearly shows the added values of the Semantic Web technologies on gaining and even deriving knowledge from streaming data. The benchmark provides a general framework to assess the ability of streaming RDF/SPARQL engines to support such applications. In the reminder of this section, the queries are grouped under section headings which indicate the features to be tested. Due to lack of space, we omit presenting the query implementations, which are available at [33].

4.1 Basic Pattern Matching

Q1. Get the rainfall observed once in an hour.

This is a basic but important query. It tests an engines ability to handle basic graph patterns, disjoint time windows ("once in an hour") to gain knowledge about the mostly spoken topic ("rainfall"), when talking about the weather.

4.2 Optional Pattern Matching

Q2. Get all precipitation observed once in an hour.

Although similar to Q1, this query is much more complex, because it requires returning all types of precipitation. Since the triple patterns for different kinds of precipitations maybe different, `OPTIONAL` patterns are needed to capture the possible differences. Additionally, this query exploits an engine's ability of reasoning over all instances of the class `PrecipitationObservation` and its subclasses.

4.3 ASK Query Form

Q3. Detect if a hurricane is being observed.

A hurricane has a sustained wind (for >3 hours) of at least 74 miles per hour. This query continuously monitors if the weather conditions observed in the current time window (i.e., an hour) is extreme ("a hurricane"). It also tests the engines ability to filter out the minimal amount of the streaming data to quickly compute the answer.

4.4 Overlapping Sliding Window and Historical Data

Q4. Get the average wind speed at the stations where the air temperature is >32 degrees in the last hour, every 10 minutes.

Combine values observed for multiple weather properties. This query tests the engines ability to deal with historical data that need to be (temporarily) stored. Moreover, contrary to queries for which an incoming data item can be immediately consumed and then discarded, this query tests how efficient an engine's strategy is to decide how to store historical data and for how long.

4.5 CONSTRUCT Derived Knowledge

Q5. Detect if a station is observing a blizzard.

A blizzard is a severe snow storm characterised by low temperatures, strong winds and heavy snow lasting for at least three hours. This query detects extreme weather conditions by combining multiple observed weather properties. It tests the engines ability to produce new knowledge derived by combining existing data.

4.6 Union

Q6. Get the stations that have observed extremely low visibility in the last hour.

Next to direct measurements of low visibility (<10 centimetres), heavy snowfall and rainfall (> 30 centimetres) also cause low visibility. This is a more complex example of detecting extreme weather conditions, which requires not only gaining knowledge explicitly contained in the data (i.e., visibility), but also deriving implicit knowledge from data sources (i.e., snowfall and rainfall).

4.7 Window-to-Stream Operation

Q7. Detect stations that are recently broken.

If a station suddenly stops producing (observation) data, it might be broken. Knowing the stability of the stations is an important issue, which can be deduced from absent data. This query tests the engines ability to cope with the dynamic properties that are specific for streaming data.

4.8 Aggregates

Q8. Get the daily minimal and maximal air temperature observed by the sensor at a given location.

Temperature is the most common weather condition queried. This query tests the engines' ability to aggregate data grouped by their geo-spatial properties.

4.9 Expression in SELECT Clause

Q9. Get the daily average wind force and direction observed by the sensor at a given location.

Wind is the other most commonly queried weather condition. The Beaufort Wind Force Scale[6] is an international standard to express how strong the wind is. It attaches some semantics to the bare wind speed numbers. Since this query requires wind speeds to be converted into Beaufort scales, it tests the engines ability to post process the qualified triple patterns.

4.10 Join with Static Data

Q10. Get the locations where a heavy snowfall has been observed in the last day.

This query finds places that are suitable for a ski holiday. It also tests the engines ability to join the dynamic sensor streaming data with the static sensor metadata.

4.11 Subquery

Q11. Detecting if a station is producing significantly different observation values than its neighbouring stations.

[6] http://en.wikipedia.org/wiki/Beaufort_scale

Detecting malfunctioning sensors is an important issue in all sensor systems. If two sensor stations are located close (denoted by `hasLocatedNearRel`) to the same location, the sensors are neighbours of each other and they should observe similar weather conditions, otherwise, a sensor might be malfunctioning. This query tests the engines ability to compute complex subqueries.

4.12 Property Path Expressions

This group of queries tests the engines ability to derive knowledge from multiple interlinked data sets using Property Path expressions. In particular, the queries require computing paths with arbitrary lengths for the `parentFeature` relationship, and computing alternatives for the name of the resulting places.

Q12. Get the hourly average air temperature and humidity of large cities.

To analyse air pollution in large cities, one might want to know if the temperature is higher during the rush hours in such cities. This query requires using the GeoNames data set to find large cities, i.e., population > 15000, and use the `hasLocatedNearRel` property in the sensor ontology [27] to find sensors located in or near to these cities.

Q13. Get the shores in Florida, US where a strong wind, i.e., the wind force is between 6 and 9, has been observed in the last hour.

This query finds shores in Florida, US, where one can go windsurfing now. It requires first reasoning over the `parentADM{1,2,3,4}` and `parentFeature` properties of the GeoNames ontology to find the shores in Florida, US; and then using the `hasLocatedNearRel` property in the sensor ontology to find sensors located near to these shores.

Q14. Get the airport(s) located in the same city as the sensor that has observed extremely low visibility in the last hour.

This query triggers an alarm if a dangerous weather condition has been observed. It requires using the GeoNames data set and the `hasLocatedNearRel` property in the sensor ontology to find airport(s) and sensors located in the same city.

4.13 Ontology-Based Reasoning

This group of queries exploit the engines ability to apply reasoning, using the properties `rdfs:subClassOf` and `owl:sameAs`, over the ontologies of the interlinked data sets.

Q15. Get the locations where the wind speed in the last hour is higher than a known hurricane.

By comparing an observed value with historical values, we can detect extreme weather conditions. This query requires reasoning over `rdfs:subClassOf` to find all known hurricanes in the system.

Q16. Get the heritage sites that are threatened by a hurricane.

We want to trigger an alarm if a dangerous weather condition has been observed. This query requires using a Property Path expression with an arbitrary length path to

Table 3. Results of the functional evaluation. The ticks indicate queries supported by an engine. Uppercase letters are abbreviations of features required by a query that are not supported by a particular system. The abbreviations are defined as the following: **A**sk; **D**stream; **G**roup by and aggregations; **IF** expression; **N**egation; **P**roperty **P**ath; and **S**tatic **D**ataset. The '/' symbol means 'or', i.e., the query would work if one of the listed features is available. The ',' symbol means 'and', i.e., all listed features are needed by the query.

	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q12	Q13	Q14	Q15	Q16	Q17
SPARQL$_{\text{Stream}}$	✓	PP	A	G	G	✓	✓	G	G,IF	SD	SD	PP,SD	PP,SD	PP,SD	PP,SD	PP,SD	PP,SD
CQELS	✓	PP	A	✓	✓	✓	D/N	✓	IF	✓	✓	PP	PP	PP	PP	PP	PP
C-SPARQL	✓	PP	A	✓	✓	✓	D	✓	IF	✓	✓	PP	PP	PP	PP	PP	PP

find all heritages sites in the DBpedia data set; then reasoning using `owl:sameAs` to link a heritage to a geographical instance described by the GeoNames data; and finally using the GeoNames data set and the `hasLocatedNearRel` property in the sensor ontology to find sensors located close to the monuments.

Q17. Estimate the damage where a hurricane has been observed.
A first thing we want to know after a natural disaster is the damage it brings. This can be estimated by consulting the damage brought by similar disasters that have happened before in the same/nearby area. This query requires using the DBpedia data set to find the damages caused by earlier hurricanes in the same/nearby area as the sensor that has observed a hurricane.

5 Implementation and Evaluation

As streaming RDF/SPARQL processing is a new topic with less than a decade of history, the proposed systems are mostly in a beginning stage of development. During this phase, one of the most important issues is to assess the effectiveness of the proposed systems. We have complemented our work on SRBench with a functional evaluation on three leading strRS systems, i.e., SPARQL$_{\text{Stream}}$ [11], CQELS [25] and C-SPARQL [6], that address strRS processing from different angles. In the evaluation, we seek answers to the following questions: Do they provide an adequate set of functionalities that are needed by the streaming applications? Do they sufficiently reveal the additional value of RDF/SPARQL for streaming data processing? Furthermore, do they provide any additional interesting functionalities, which help the users to distinguish one system from the others? The results presented in this section are meant to give a first insight into the state-of-the-art of the strRS engines and highlight several pros and cons of different approaches.

All three systems come with a streaming SPARQL language proposal and a query processing engine. The queries are implemented according to the syntax specified in [11], [25] and [6] for SPARQL$_{\text{Stream}}$, CQELS and and C-SPARQL, respectively. To execute the queries, we downloaded the latest SPARQL$_{\text{Stream}}$[7], CQELS (Aug. 2011)[8] and C-SPARQL 0.7.4[9]. All implementing queries can be found in the SRBench wiki page [33]. An overview of the evaluation results is shown in Table 3. A tick indicates that the engine is able to process a particular query. For each query that cannot be processed by a certain engine, we denote the main missing feature(s) that cause the query to fail. The abbreviations are defined in the table caption.

The results of the evaluation are fairly close to our expectation. In general, all three engines support basic SPARQL features (i.e., the graph pattern matching features, solution modifiers and the `SELECT` and `CONSTRUCT` query forms discussed in Section 4) over time-based windows of streaming data. The main issue we have identified from the evaluation is that all three engines' abilities of dealing with the new features introduced by SPARQL 1.1 are rather limited. So, for instance, seven out of seventeen queries cannot be executed, because the Property Path expressions are not supported by any of the tested engines. We regard the Property Path expressions as one of the most important features of SPARQL 1.1, because it provides flexible ways to navigate through RDF graphs and facilitates reasoning over various graph patterns. This feature does not exist in other query languages, e.g., SQL and XQuery, that are not primarily designed to query graph data. The lacking of support for the SPARQL 1.1 features is most probably due to the freshness of the SPARQL 1.1 language. However, we would like to emphasise that the advanced SPARQL 1.1 features are the distinguishing factors of the usefulness of one system from others for streaming RDF applications.

Several more remarks can be made from Table 3. The lack of support for the `ASK` query form needed by Q3 is unexpected, since it is an easy to implement feature. This might be caused by that it does not have research values, but in real-world scenario's, such as the one used by SRBench, the users often start with asking for basic information that can be just as simple as "is it raining?". Q3 can also be implemented using the `IF` function of SPARQL 1.1, but it is available on none of the tested engines.

Q7 can be most easily implemented using the window-to-stream operator `DSTREAM`, which can detect data items that have been deleted since the previous window, which is exactly what this query asks. Although all three systems provide ways to produce new data streams from the results of a continuous query, SPARQL$_{\text{Stream}}$ is the only streaming SPARQL extension that support `DSTREAM`. Q7 can also be implemented by querying the same stream twice with different window definition and then using the `NOT EXISTS` expression of SPARQL 1.1 to test the absence of a graph pattern from the previous time interval in the current time interval. Since CQELS allows defining different time windows for the same stream, this query can be expressed in the CQELS language, but query execution failed because the CQELS engine does not support `NOT EXISTS` yet. C-SPARQL does not allow query the same stream more than once, and SPARQL 1.1 does not define arithmetic functions over the date/time data type, it is not possible to express the query in C-SPARQL.

[7] `http://code.google.com/p/semanticstreams/source/checkout`
[8] `http://code.google.com/p/cqels/`
[9] `http://streamreasoning.org/download`

The number of functionalities currently supported by SPARQL$_{\text{Stream}}$ is somewhat less than those supported by CQELS and C-SPARQL. As the `GROUP BY` and aggregations functions are still work in progress in SPARQL$_{\text{Stream}}$, it causes four more queries (i.e., Q4, Q5, Q8 and Q9) to fail. Moreover, the development of the SPARQL$_{\text{Stream}}$ engine has so far concentrated on supporting streaming data, but not both streaming and static data. This is one of the main reasons that the queries Q10 – 17 cannot be run on SPARQL$_{\text{Stream}}$. Enabling queries on both streaming and static data sets is an ongoing subproject of PlanetData[10].

Little work has been done on enabling reasoning over streaming data. C-SPARQL is the only testing system that supports reasoning based on simple RDF entailment. SPARQL$_{\text{Stream}}$ and CQELS currently do not tackle the problem of reasoning, because SPARQL$_{\text{Stream}}$ targets at enabling ontology based access to streaming data, while CQELS concentrates on building a strRS engine from scratch. So, next to the Property Path expressions, the ability to apply reasoning is another distinguishing factor.

The overall conclusion of our evaluation is that there is no single best system yet, even thought the SPARQL$_{\text{Stream}}$ engine supports fewer queries than CQELS and C-SPARQL. Both the SPARQL 1.1 language and the streaming RDF/SPARQL engines have been introduced only recently. As the time passes, we expect the strRS engines to gradually provide a richer set of functionalities.

Although this work focuses on a functional evaluation, we propose a number of metrics that should be used for a performance evaluation using SRBench. *Correctness*: the query results must be validated, taking into account possible variations in ordering, and possibly multiple valid results per query. The validation results should be expressed in terms of *precision* and *recall*. *Throughput*: the maximal number of incoming data items a strRS engine is able to process per time unit. *Scalability*: how does the system reacts to increasing number of incoming streams and continuous queries to be processed. *Response time*: the minimal elapsed time between a data item entering the system and being returned as output of a query. Note that response time is mainly relevant for queries allowing immediate query results upon receiving of a data item.

6 Related Work

The Linear Road Benchmark [3] is the only publicly available benchmark developed for evaluating traditional data stream engines. The benchmark simulates a traffic management scenario where multiple cars are moving on multiple lanes and on multiple different roads. The system to be tested is responsible to monitor the position of each car, and continuously calculates and reports to each car the tolls it needs to pay and whether there is an accident that might affect it. In addition, the system needs to continuously maintain historical data, as it is accumulated, and report to each car the account balance and the daily expenditure. Linear Road is a highly challenging and complicated benchmark due to the complexity of the many requirements. It stresses the system and tests various aspects of its functionality, e.g., window-based queries, aggregations, various kinds of complex join queries; theta joins, self-joins, etc. It also requires the ability to evaluate not only continuous queries on the stream data, but also historical queries

[10] `http://planet-data-wiki.sti2.at/`

on past data. The system should be able to store and later query intermediate results. All these features are also addressed in SRBench. Additionally, SRBench includes Semantic Web specified features, such as inter liked heterogeneous data sets, exploration of RDF graphs and reasoning.

With the growth and availability of many systems supporting RDF/SPARQL, increasing efforts have been made in developing benchmarks for evaluating the performance of RDF stores. The most representative and widely used RDF benchmarks are the Lehigh University Benchmark (LUBM) [21], the Berlin SPARQL Benchmark (BSBM) [8], and the SPARQL Performance Benchmark (SP^2Bench) [30]. LUBM, one of the first RDF benchmarks, is built over a university domain in order to mainly evaluate the reasoning capability and inference mechanism of OWL (Web Ontology Language) Knowledge Base Systems. SP^2 Bench uses DBLP[11] as its domain and generates the synthetic data set mimicking the original DBLP data. Currently, BSBM is probably the most popular RDF/SPARQL benchmark that is built around an e-commerce use case where products are offered by various vendors and get the reviews from various customers in different review sites. However, these benchmarks are mostly relational-like, lack heterogeneity or are limited in representing realistic skewed data distributions and correlations. No new features of SPARQL 1.1, such as property path expression have been addressed in these benchmarks. Besides, although one advantage of RDF is the flexibility in sharing and integrating linked open knowledge bases, existing benchmarks solely work with one generated data set without exploiting the knowledge from other linked open data such as DBpedia [17]. Finally, none of the existing benchmarks provides reasoning tasks, a distinguish feature of Semantic Web technology. SRBench has advanced the state-of-the-art of RDF benchmarks by addressing all these features that are hitherto absent.

7 Conclusion

We have introduced SRBench, the first general purpose streaming RDF/SPARQL benchmark, that has been primarily designed to assess the abilities of streaming RDF/SPARQL processing engines in applying Semantic Web technologies on streaming data. The benchmark has been designed based on an extensive study of the state-of-the-art techniques in both the data stream management systems and the strRS processing engines. This ensures that we capture all important aspects of strRS processing in the benchmark.

Motivated by the study of [17], we have carefully chosen three real-world data sets from the LOD cloud to be used in the benchmark. The benchmark contains a concise, yet comprehensive set of queries which covers the major aspects of streaming SPARQL query processing, ranging from simple graph pattern matching queries to queries with complex reasoning tasks. The main advantages of applying Semantic Web technologies on streaming data include providing better search facilities by adding semantics to the data, reasoning through ontologies, and integration with other data sets. The ability of a strRS engine to process these distinctive features is accessed by the benchmark with queries that apply reasoning not only over the streaming sensor data, but also over the metadata and even other data sets in the LOD cloud. We have complemented

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

our work on SRBench with a functional evaluation of the benchmark on three currently leading streaming RDF/SPARQL engines. The evaluation shows that the functionalities provided by the tested engines are generally limited to basic RDF/SPARQL features over streaming data. There is no single best system yet. We believe that a streaming RDF/SPARQL engine can significantly differentiate itself from other strRS engines by providing more advanced SPARQL 1.1 features and reasoning over both streaming and static data sets.

The natural next step is to run performance and scalability evaluations on the three example strRS engines and probably even other engines. This is a ongoing work. A practical challenge in doing performance evaluation is the verification of query results, given the dynamicity of streaming data and the diversity of the implementing engines.

Acknowledgements. This works was supported by grants from the European Union's 7th Framework Programme (2007-2013) provided for the projects PlanetData (GA no. 257641) and LOD2 (GA no. 257943). We would like to give special thanks to Phuoc-Danh Le for his help on using CQELS.

References

1. Anicic, D., Fodor, P., Rudolph, S., Stojanovic, N.: EP-SPARQL: a unified language for event processing and stream reasoning. In: WWW 2011, pp. 635–644 (2011)
2. Arasu, A., Babu, S., Widom, J.: CQL: A Language for Continuous Queries over Streams and Relations. In: Lausen, G., Suciu, D. (eds.) DBPL 2003. LNCS, vol. 2921, pp. 1–19. Springer, Heidelberg (2004)
3. Arasu, A., et al.: Linear Road: A Stream Data Management Benchmark. In: Proc. of the 30th VLDB Conference, Toronto, Canada, pp. 480–491 (2004)
4. Arenas, M., Conca, S., Pérez, J.: Counting Beyond a Yottabyte, or how SPARQL 1.1 Property Paths will Prevent Adoption of the Standard. In: WWW (2012)
5. Balazinska, M., et al.: Data Management in the Worldwide Sensor Web. IEEE Pervasive Computing 6(2), 30–40 (2007)
6. Barbieri, D.F., Braga, D., Ceri, S., Della Valle, E., Grossniklaus, M.: Querying RDF Streams with C-SPARQL. SIGMOD Record 39(1), 20–26 (2010)
7. Berners-Lee, T.: Linked Data - Design Issues (2009), http://www.w3.org/DesignIssues/LinkedData.html
8. Bizer, C., Schultz, A.: The Berlin SPARQL Benchmark. Int. J. Semantic Web Inf. Syst. 5(2), 1–24 (2009)
9. 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)
10. Bouillet, E., Feblowitz, M., Liu, Z., Ranganathan, A., Riabov, A., Ye, F.: A Semantics-Based Middleware for Utilizing Heterogeneous Sensor Networks. In: Aspnes, J., Scheideler, C., Arora, A., Madden, S. (eds.) DCOSS 2007. LNCS, vol. 4549, pp. 174–188. Springer, Heidelberg (2007)
11. 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)

12. CKAN - the Data Hub, http://thedatahub.org/
13. Corcho, O., et al.: Characterisation mechanisms for unknown data sources. EU Project PlanetData (FP7-257641), Deliverable 1.1 (2011)
14. Corcho, O., García-Castro, R.: Five challenges for the semantic sensor web. Semantic Web 1(1), 121–125 (2010)
15. DBpedia, http://wiki.dbpedia.org/
16. Della Valle, E., et al.: It's a Streaming World! Reasoning upon Rapidly Changing Information. IEEE Intelligent Systems 24(6), 83–89 (2009)
17. Duan, S., Kementsietsidis, A., Srinivas, K., Udrea, O.: Apples and Oranges: A Comparison of RDF Benchmarks and Real RDF Datasets. In: SIGMOD (2011)
18. GeoNames Ontology, http://www.geonames.org/ontology/
19. Gray, J.: The Benchmark Handbook for Database and Transaction Systems. Morgan Kaufmann (1993)
20. Groppe, S., et al.: A SPARQL Engine for Streaming RDF Data. In: SITIS (2007)
21. 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)
22. Harris, S., Seaborne, A.: SPARQL 1.1 Query Language. W3C Working Draft, World Wide Web Consortium (January 05, 2012), http://www.w3.org/TR/sparql11-query/
23. Hey, T., Tansley, S., Tolle, K. (eds.): The Fourth Paradigm: Data-Intensive Scientific Discovery. Microsoft Research (October 2009)
24. Hoeksema, J.: A Parallel RDF Stream Reasoner and C-SPARQL Processor Using the S4 Framework. Master's thesis, VU University, Amsterdam, The Netherlands (October 2011)
25. 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)
26. Le-Phuoc, D., Hauswirth, M.: Linked open data in sensor data mashups. In: Proceedings of the 2nd International Workshop on Semantic Sensor Networks (SSN 2009), pp. 1–16 (2009)
27. LinkedSensorData, http://wiki.knoesis.org/index.php/LinkedSensorData
28. Pérez, J., et al.: Semantics and Complexity of SPARQL. ACM TODS 34(3), 1–45 (2009)
29. Prud'hommeaux, E., Seaborne, A.: SPARQL Query Language for RDF. W3C Recommendation, World Wide Web Consortium (January 15, 2008)
30. Schmidt, M., et al.: SP^{2}Bench: A SPARQL Performance Benchmark. In: ICDE (2009)
31. Sequeda, J., Corcho, O.: Linked stream data: A position paper. In: Proceedings of Semantic Sensor Networks, pp. 148–157 (2009)
32. Sheth, A.P., et al.: Semantic Sensor Web. IEEE Internet Computing 12(4), 78–83 (2008)
33. SRBench wiki, http://www.w3.org/wiki/SRBench
34. The Linking Open Data cloud diagram, http://richard.cyganiak.de/2007/10/lod/
35. Walavalkar, O., et al.: Streaming Knowledge Bases. In: SSWS (2008)
36. Whitehouse, K., Zhao, F., Liu, J.: Semantic Streams: A Framework for Composable Semantic Interpretation of Sensor Data. In: Römer, K., Karl, H., Mattern, F. (eds.) EWSN 2006. LNCS, vol. 3868, pp. 5–20. Springer, Heidelberg (2006)
37. Zhang, Y., et al.: Benchmarking RDF Storage Engines. EU Project PlanetData, Deliverable 1.2 (2011)

Scalable Geo-thematic Query Answering

Özgür Lütfü Özçep and Ralf Möller

Institute for Software Systems (STS)
Hamburg University of Technology
Hamburg, Germany
{oezguer.oezcep,moeller}@tu-harburg.de

Abstract. First order logic (FOL) rewritability is a desirable feature for query answering over geo-thematic ontologies because in most geo-processing scenarios one has to cope with large data volumes. Hence, there is a need for combined geo-thematic logics that provide a sufficiently expressive query language allowing for FOL rewritability. The DL-Lite family of description logics is tailored towards FOL rewritability of query answering for unions of conjunctive queries, hence it is a suitable candidate for the thematic component of a combined geo-thematic logic. We show that a weak coupling of DL-Lite with the expressive region connection calculus RCC8 allows for FOL rewritability under a spatial completeness condition for the ABox. Stronger couplings allowing for FOL rewritability are possible only for spatial calculi as weak as the low-resolution calculus RCC2. Already a strong combination of DL-Lite with the low-resolution calculus RCC3 does not allow for FOL rewritability.

Keywords: FOL rewritability, description logics, region connection calculus, qualitative spatial reasoning, GIS, combined logic.

1 Introduction

Query answering over a database becomes far more difficult if the extensional knowledge in the database is extended by constraints in an ontology. The reason is that a database plus an ontology may have many different models, hence ontology based query answering has to compute the answers w.r.t. to all models and build their intersection (*certain answer* semantics). But in some cases—when using a lightweight logic like DL-Lite for the representation of the ontology and a restricted query language like unions of conjunctive queries—query answering w.r.t. an ontology can be reduced to model checking. This is formalized by the notion of *FOL (first order logic) rewritability*: a given query can be rewritten into a FOL query in which the intensional knowledge of the ontology is captured. Though the rewritten queries may become exponentially bigger than the original ones, there exist optimizations based on semantic indexes which encode entailed knowledge of the terminological part of the ontology [15]. So, FOL rewritability means a benefit.

DL-Lite per se [3] is not sufficient for use in scenarios of geographic information processing, as these demand, among others, the representation and deduction over spatial concepts. Though constraint-based spatial reasoning [14] offers a well developed and well proven theory for spatial domains, it does not fill in the need for a system that combines reasoning over a spatial and a non-spatial (thematic) domain. Though constraint databases [7] are good candidate frameworks for reasoning over a mixed domain of geo-thematic objects, the investigations on constraint databases so far did not incorporate terminological reasoning in the OBDA (ontology based data access) paradigm. But even in case of related work which equally considers spatial and thematic reasoning [5], [17], [8], it is not aimed at FOL rewritability. Hence, there is still a need for investigating combinations of logics that, on the one hand, are sufficiently expressive to match the representation requirements in geographical information processing and that, on the other hand, allow for computationally feasible (in particular FOL rewritable) satisfiability checking and query answering.

We would like to illustrate the use of the logics of this paper by a simple scenario in which an engineering bureau plans additional parks in a city [10]. Assume, the bureau has stored geographical data in some database (DB) and declares relevant concepts in the terminological part of his knowledge base, the TBox. The engineer gives necessary conditions for a concept *Park+Lake* which is a park containing a lake that touches it from within, i.e., using the terminology of the region connection calculus (RCC) [11], the lake is a tangential proper part of the park. Similarly, a necessary condition for the concept *Park4Playing* is given which is a park containing a playing ground (for children) that is a tangential proper part.

We assume that the data are mapped to a logical pendant of the DB called the ABox (assertional box). In particular the data should generate the fact that there is an object a which is both a park with a lake and with a playing area, that is $\mathit{Park{+}Lake}(a)$ and $\mathit{Park4Playing}(a)$ are contained in the ABox. But the location of a is not known. Think of a as an object whose architectural design is determined but the place where a is going to be localized is not determined yet.

Now, the engineering bureau asks for all parks with lakes and playing areas such that the playing area is not contained as island in the lake. These kinds of parks can be thought of as secure as the playing ground can be directly reached from the park (without a bridge). All objects that fall into the answer set of this query w.r.t. to the TBox and the data can have one of the configurations A to C illustrated in Figure 1 (and many more) but are not allowed to have the configuration D. The object a has to be in the answer set to the original query as the TBox together with the ABox and some deduction on the spatial configuration implies that a is an object which excludes the island configuration D. Remember that a is "abstract" in so far as its geographical location is not known. So in fact deduction is needed to see that a does not have configuration D. Later on we will formalize this example in the logic DL-Lite$^{\sqcap}_{\mathcal{F},\mathcal{R}}$(RCC8) and the query language GCQ^{+} and show that the deduction needed for the inclusion of a into the answer set can be mimicked in a perfect rewriting algorithm.

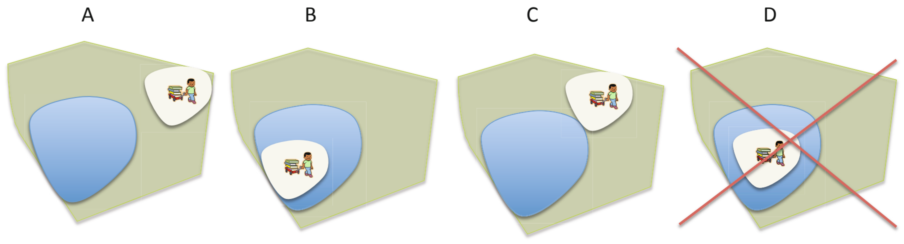


Fig. 1. (Dis-)Allowed spatial configurations for query in engineering bureau scenario

Continuing previous work [10], we investigate combinations of logics in the DL-Lite family with different members of RCC family [11], a well-known family of calculi for qualitative spatial reasoning. In previous work [10], we focussed on the FOL rewritability aspects for weak combinations of DL-Lite with RCC8; these combinations are weak in so far as they do not allow for the construction of arbitrary RCC8 constraint networks in the intensional part (TBox) of the ontology. In this paper we extend these investigations by enlarging the expressivity of DL-Lite to one that allows for concept conjunctions on the left-hand side of general inclusion axioms [1], and we give a proof including a rewriting algorithm for the main result stating that the weak combination of DL-Lite$^{\sqcap}_{\mathcal{F},\mathcal{R}}$ with RCC8 allows for FOL rewriting w.r.t. to the query language GCQ^+.

Moreover, in this paper, we consider strong combinations of DL-Lite with the weaker RCC fragments RCC3 and RCC2, and prove that DL-Lite$^{\sqcap}_{\mathcal{F},\mathcal{R}}$(RCC3) does not allow for FOL rewritability of satisfiability checking while the weaker DL-Lite$^{\sqcap}_{\mathcal{F},\mathcal{R}}$(RCC2) does [9].

The paper is structured as follows. Section 2 collects technical details on the region connection calculus and the DL-Lite family of description logics. Weak combinations of DL-Lite with the region connection calculus are described in Sect. 3. In Sect. 4, the last section before the conclusion, we consider strong combinations of DL-Lite with weaker fragments of the region connection calculus.

Full proofs can be found in an extended version of this paper available under the URL http://dl.dropbox.com/u/65078815/oezcepMoellerISWC2012Ext.pdf.

2 Logical Preliminaries

We recapitulate the main logical notation and concepts used in this paper; the region connection calculus and DL-Lite.

2.1 The Region Connection Calculus

We will consider different fragments of the region connection calculus [11] as potential candidates for the spatial logic to be combined with DL-Lite. Randell and colleagues' axiom system [11] is based on a primitive binary relation C intended to denote a connectedness relation which is specified to be reflexive and symmetric. On the basis of C other binary relations between regions which are called *base relations* are explained. One set of base relations is the set $\mathcal{B}_{RCC8}$, which is the main component of the most expressive region connection calculus RCC8. The base relations of $\mathcal{B}_{RCC8}$ and their intended meanings are given as follows: $\mathcal{B}_{RCC8} = \{\mathsf{DC}$ (disconnected), EC (externally connected), EQ (equal), PO (partially overlapping), NTPP (non-tangential proper part), TPP (tangential proper part), NTPPi (inverse of NTPP), TPPi (inverse of TPP)$\}$. We skip the concrete definitions of the base relations by the connectedness relation C (see, e.g., [13, p. 45]), as we—in contrast to the axiom system of Randell and colleagues—consider the following axiom system schema Ax_{RCCi}, which directly specifies the properties of the base relations in $\mathcal{B}_{RCCi}$.

Definition 1 (Axiom system schema Ax_{RCCi}). *For all $i \in \{2,3,5,8\}$ the axiom set Ax_{RCCi} contains the following axioms:*

$$\{\forall x, y. \bigvee_{r \in \mathcal{B}_{RCCi}} r(x,y)\} \cup \qquad \textit{(joint exhaustivity)}$$
$$\{\forall x, y. \bigwedge_{r_1, r_2 \in \mathcal{B}_{RCCi}, r_1 \neq r_2} r_1(x,y) \rightarrow \neg r_2(x,y)\} \cup \qquad \textit{(pairwise disjointness)}$$
$$\{\forall x, y, z. r_1(x,y) \wedge r_2(y,z) \rightarrow r_3^1(x,z) \vee \cdots \vee r_3^k(x,z) \mid r_1 ; r_2 = \{r_3^1, \ldots, r_3^k\}\}$$
$$\textit{(weak composition axioms)}$$

For $i \in \{3,5,8\}$ additionally the axiom $\forall x \mathsf{EQ}(x,x)$ (reflexivity of EQ) is contained. For $i = 2$ the axiom $\forall x \mathsf{O}(x,x)$ (reflexivity of O) is contained.

In particular, the axioms state the JEPD-property of the base relations (each pair of regions x, y is related over exactly one base relation) and describe the (weak) composition of two base relations (denoted by ;) according to the composition table for RCCi. With the composition of two base relations, in most cases, only indefinite knowledge of spatial configurations follows. The spatial configuration $r_1(x,z) \vee \cdots \vee r_n(x,z)$ for base relations r_j in $\mathcal{B}_{RCCi}$ is also written as $\{r_1, \ldots, r_n\}(x,z)$, and the set $\{r_1, \ldots, r_n\}$ is called a general RCCi relation. Let Rel_{RCCi} be the set of all 2^i general RCCi relations. An RCCi *(constraint) network* consists of assertions of the form $\{r_1, \ldots, r_n\}(x,y)$.

We mention here the composition table for the low resolution logics RCC2 and RCC3. Their base relations are given by the sets $\mathcal{B}_{RCC3} = \{\mathsf{DR}, \mathsf{EQ}, \mathsf{ONE}\}$ and $\mathcal{B}_{RCC2} = \{\mathsf{DR}, \mathsf{O}\}$, and their weak compositions are defined as shown in Fig. 2. The discreteness relation DR is the same as $\{\mathsf{DC}, \mathsf{EC}\}$, the overlapping-but-not-equal relation ONE is equal to $\{\mathsf{PO}, \mathsf{NTPP}, \mathsf{TPP}, \mathsf{NTPPi}, \mathsf{TPPi}\}$ and the overlapping relation O is given by $\{\mathsf{ONE}, \mathsf{EQ}\}$.

;	DR	O
DR	$\mathcal{B}_{RCC2}$	$\mathcal{B}_{RCC2}$
O	$\mathcal{B}_{RCC2}$	$\mathcal{B}_{RCC2}$

;	DR	ONE	EQ
DR	$\mathcal{B}_{RCC3}$	{DR, ONE}	DR
ONE	{DR, ONE}	$\mathcal{B}_{RCC3}$	ONE
EQ	DR	ONE	EQ

Fig. 2. Composition tables for RCC2 and RCC3

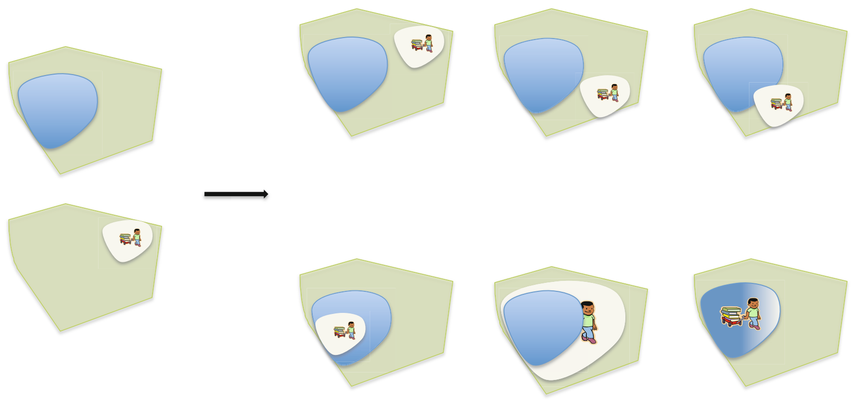


Fig. 3. Illustration for composition entry $\mathsf{tpp};\mathsf{tppi}$

Note that in the definitions of the base relations (of RCC3 and RCC2) we followed the author of [16] and not [4]. But the composition tables for both definitions are identical. For the composition tables of RCC5 and RCC8 confer [14, p. 45]. As an example entry for RCC8, which is relevant for the engineering bureau scenario, we mention the table entry for the pair $(\mathsf{tpp}, \mathsf{tppi})$: $\mathsf{tpp};\mathsf{tppi} = \{\mathsf{dc}, \mathsf{ec}, \mathsf{po}, \mathsf{tpp}, \mathsf{tppi}, \mathsf{eq}\}$ which is described in Ax_{RCC8} by $\forall x, y, z.\mathsf{tpp}(x, y) \wedge \mathsf{tppi}(y, z) \rightarrow \{\mathsf{dc}, \mathsf{ec}, \mathsf{po}, \mathsf{tpp}, \mathsf{tppi}, \mathsf{eq}\}(x, z)$. In case of the engineering bureau scenario from the introduction the constraint of this composition entry is demonstrated for x being the lake, y being the park and z being the playing ground.

2.2 DL-Lite

The family of DL-Lite description logics [3] is an appropriate candidate for the thematic component of the envisioned geo-thematic logic as it offers computationally feasible satisfiability checking and query answering over ontologies and data stored in a relational database. More concretely, satisfiability checking and query answering (for unions of conjunctive queries) are FOL rewritable. In this paper, we mainly deal with a member of the extended DL-Lite family DL-Lite$^{\sqcap}_{\mathcal{F},\mathcal{R}}$; it allows for functional roles, role hierarchies, role inverses and conjunction of basic concepts on the left-hand side of GCIs (general concept inclusions). The syntax is given in Def. 2. The semantics of this logic is defined in the usual way—but imposing the unique name assumption (UNA).

Definition 2 (DL-Lite$^{\sqcap}_{\mathcal{F},\mathcal{R}}$). *Let RN be the set of role symbols and $P \in RN$, CN be a set of concept symbols and $A \in CN$, $Const$ be a set of individual constants and $a, b \in Const$.*

$$R \longrightarrow P \mid P^{-} \quad B \longrightarrow A \mid \exists R \quad C_l \longrightarrow B \mid C_l \sqcap B \quad C_r \longrightarrow B \mid \neg B$$

TBox$^{*)}$: $\quad C_l \sqsubseteq C_r, (\text{funct } R), R_1 \sqsubseteq R_2$

ABox: $\quad A(a), R(a, b)$

**) Restriction: If R occurs in a functionality axiom, then R and its inverse do not occur on the right-hand side of a role inclusion axiom $R_1 \sqsubseteq R_2$.*

FOL rewritability also holds for the logic DL-Lite$^{\sqcap}_{\mathcal{F},\mathcal{R}}$ which follows from the corresponding FOL rewritability results for the Datalog extension Datalog$^{\pm}$ [2]. We recapitulate the technical notions needed for defining FOL rewritability. An ontology $\mathcal{O}$ is a tuple $(Sig, \mathcal{T}, \mathcal{A})$, with a signature Sig (i.e., a set of concept symbols, role symbols and constants also denoted by $Sig(\mathcal{O})$), with a TBox $\mathcal{T}$, and with an ABox $\mathcal{A}$. An *FOL query* $Q = \psi(\boldsymbol{x})$ is an FOL formula $\psi(\boldsymbol{x})$ with free variables $\boldsymbol{x}$ called *distinguished variables.* If $\boldsymbol{x}$ is empty, the query is called *boolean.* Let $\boldsymbol{a}$ be a vector of constants from $Sig(\mathcal{O})$. The set of answers w.r.t. $\mathcal{I}$ is defined by $Q^{\mathcal{I}} = \{\boldsymbol{d} \in (\Delta^{\mathcal{I}})^n \mid \mathcal{I}_{[\boldsymbol{x} \mapsto \boldsymbol{d}]} \models \psi(\boldsymbol{x})\}$. (We use $Q^{\mathcal{I}}$ later on for a specific model $\mathcal{I}$, namely a Herbrand model.) The set of certain answers w.r.t. to $\mathcal{O}$ is defined by $cert(Q, \mathcal{T} \cup \mathcal{A}) = \{\boldsymbol{a} \mid \mathcal{T} \cup \mathcal{A} \models \psi[\boldsymbol{x}/\boldsymbol{a}]\}$. A *conjunctive query (CQ)* is a FOL query in which $\psi(\boldsymbol{x})$ is an existentially quantified conjunction of atomic formulas $at(\cdot)$, $\psi(\boldsymbol{x}) = \exists \boldsymbol{y} \bigwedge_i at_i(\boldsymbol{x}, \boldsymbol{y})$. A *union of conjunctive queries (UCQ)* is a disjunction of CQs, i.e., a formula of the form $\exists \boldsymbol{y_1} \bigwedge_{i_1} at_{i_1}(\boldsymbol{x}, \boldsymbol{y}_1) \vee \cdots \vee \exists \boldsymbol{y_n} \bigwedge_{i_n} at_{i_n}(\boldsymbol{x}, \boldsymbol{y}_n)$. We conceive a UCQ as a set of CQs. The existential quantifiers in UCQs are interpreted in the same way as for FOL formulas (natural domain semantics) and not w.r.t. a given set of constants mentioned in the signature (active domain semantics).

Let $DB(\mathcal{A})$ be the minimal Herbrand model of $\mathcal{A}$. *Checking the satisfiability of ontologies is FOL rewritable* iff for all TBoxes $\mathcal{T}$ there is a boolean FOL query $Q_{\mathcal{T}}$ s.t. for all ABoxes $\mathcal{A}$: the ontology $\mathcal{T} \cup \mathcal{A}$ is satisfiable iff $DB(\mathcal{A}) \not\models Q_{\mathcal{T}}$. *Answering queries from a subclass $\mathcal{C}$ of FOL queries w.r.t. to ontologies is FOL rewritable* iff for all TBoxes $\mathcal{T}$ and queries $Q = \psi(\boldsymbol{x})$ in $\mathcal{C}$ there is a FOL query $Q_{\mathcal{T}}$ such that for all ABoxes $\mathcal{A}$ it is the case that $cert(Q, \mathcal{T} \cup \mathcal{A}) = Q_{\mathcal{T}}^{DB(\mathcal{A})}$. For DL-Lite, FOL-rewritability can be proved w.r.t. to satisfiability as well as w.r.t. answering UCQs [3, Thm 4.14, Thm 5.15].

The rewritability results are proved with the so called chase construction known from database theory. The idea of the chase construction is to repair the ABox with respect to the constraints formulated in the TBox. If, e.g., the TBox contains the axiom $A_1 \sqsubseteq A_2$ and the ABox contains $A_1(a)$ but not $A_2(a)$, then it is enriched by the atom $A_2(a)$. This procedure is applied stepwise to yield a sequence of ABoxes S_i starting with the original ABox as S_0. The resulting set of ABox axioms $\bigcup S_i$ may be infinite but induces a canonical model $can(\mathcal{O})$ for the ABox and the TBox axioms being used in the chasing process (see below). We will summarize the chase construction for DL-Lite.

Let $\mathcal{T}$ be a DL-Lite TBox, let $\mathcal{T}_p$ be the subset of positive inclusion (PI) axioms in $\mathcal{T}$ (no negation symbol allowed) and let $\mathcal{A}$ be an ABox and $\mathcal{O} = \mathcal{T} \cup \mathcal{A}$. Chasing will be carried out with respect to PIs only. Let $S_0 = \mathcal{A}$. Let S_i be the set of ABox axioms constructed so far and α be a PI axiom in $\mathcal{T}_p$. Let α be of the form $A_1 \sqsubseteq A_2$ and let $\beta \in S_i$ (resp. $\beta \subseteq S_i$) be an ABox axiom (resp. set of ABox axioms). The PI axiom α is called applicable to β if β is of the form $A_1(a)$ and $A_2(a)$ is not in S_i. The applicability of other PI axioms of the form $B \sqsubseteq C$ is defined similarly [3, Def. 4.1, p. 287]. If the left-hand side of the PI is a conjunction of base concepts, e.g., if the PI is of the form $A_1 \sqcap \cdots \sqcap A_n \sqsubseteq A_0$, and if β is $\{A_1(a), \ldots, A_n(a)\}$ and $A_0(a)$ is not in S_i, then PI is applicable to β.

As there may be many possible applications of PI axioms to atoms and sets of atoms, one has to impose an order on the TBox axioms and the (finite) subsets of the ABox. So we assume that all strings over the signature $Sig(\mathcal{O})$ of the ontology and some countably infinite set of new constants C_{ch} are well ordered. Such a well ordering exists and has the order type of the natural numbers $\mathbb{N}$. This ordering is different from the one of [3]; but it can also be used also for infinite ABoxes and it can handle concept conjunction. If there is a PI axiom α applicable to an atom β in S_i, one takes the minimal pair (α, β) with respect to the ordering and produces the next level $S_{i+1} = S_i \cup \{\beta_{new}\}$; here β_{new} is the atom that results from applying the chase rule for (α, β) as listed in Def. 3. The primed constants are the chasing constants from C_{ch}.

Definition 3 (Chasing rules for DL-Lite$^{\sqcap}_{\mathcal{F},\mathcal{R}}$)

If $\alpha = A_1 \sqsubseteq A_2$ *and* $\beta = A_1(a)$ *then* $\beta_{new} = A_2(a)$
If $\alpha = A_1 \sqsubseteq \exists R$ *and* $\beta = A_1(a)$ *then* $\beta_{new} = R(a, a')$
If $\alpha = \exists R \sqsubseteq A$ *and* $\beta = R(a, b)$ *then* $\beta_{new} = A(a)$
If $\alpha = \exists R_1 \sqsubseteq \exists R_2$ *and* $\beta = R_1(a, b)$ *then* $\beta_{new} = R_2(a, a')$
If $\alpha = R_1 \sqsubseteq R_2$ *and* $\beta = R_1(a, b)$ *then* $\beta_{new} = R_2(a, b)$
If $\alpha = A_1 \sqcap \cdots \sqcap A_n \sqsubseteq A_0$ *and* $\beta = \{A_1(a), \ldots, A_n(a)\}$ *then* $\beta_{new} = A_2(a)$
(and similarly for other PIs of the form $B_1 \sqcap \cdots \sqcap B_n \sqsubseteq C$*)*

The chase is defined by $chase(\mathcal{O}) = chase(\mathcal{T}_p \cup \mathcal{A}) = \bigcup_{i \in \mathbb{N}} S_i$. The canonical model $can(\mathcal{O})$ is the minimal Herbrand model of $chase(\mathcal{O})$. The canonical model $can(\mathcal{O})$ is a universal model of $\mathcal{T}_p \cup \mathcal{A}$ with respect to homomorphisms. In particular this implies that answering a UCQ Q w.r.t. to $\mathcal{T}_p \cup \mathcal{A}$ can be reduced to answering $Q^{can(\mathcal{O})}$ w.r.t. to $DB(\mathcal{A})$. More concretely, (some finite closure $cln(\mathcal{T})$ of) the negative inclusions axioms and the functionality axioms are only relevant for checking the satisfiability of the ontology which can be tested by a simple FOL query. We leave out the details (see the extended version of this paper).

3 Weak Combinations of DL-Lite with RCC

In this section, we extend the results concerning a weak coupling of DL-Lite with the most expressive region connection calculus fragment RCC8, which we introduced in [10], and explain its use(fulness) with a formalization of the example scenario from the introduction. This will give us the opportunity to introduce further concepts that are necessary to understand the discussions on stronger couplings of DL-Lite with the weaker region connection calculi RCC2 and RCC3.

The combination paradigm follows that of Lutz and Miličič [8] who combine $\mathcal{ALC}$ with the RCC8 and, more generally, with ω-admissible concrete domains [8, Def. 5, p. 7]. The combined logic $\mathcal{ALC}(RCC8)$ of [8] is well behaved in so far as testing concept subsumption is decidable. As we aim at FOL rewritability we have to be even more careful in choosing the right combination method.

We use an axiom set T_ω with corresponding properties of an ω-admissible domain for coupling with DL-Lite because axioms are more appropriate for rewriting investigations. The axiom sets Ax_{RCCi} will instantiate T_ω.

We recapitulate the syntax and the semantics of the constructors of [8] that are used for the coupling of the thematic and the spatial domain. A path U (of length at most 2) is defined as l for a fixed attribute l ("has location") or as $R \circ l$, the composition of the role symbol R with l. We abbreviate $R \circ l$ with $\tilde{R}$ in this paper. The usual notion of an interpretation $\mathcal{I}$ in our combined logic is slightly modified by using two separate domains $\Delta^{\mathcal{I}}$ and $(\Delta^*)^{\mathcal{I}}$. All symbols of the theory T_ω are interpreted relative to $(\Delta^*)^{\mathcal{I}}$. Let r be an RCC-relation of some RCC-fragment. That is, let be given a set of base relations $\mathcal{B}_{RCCi}$ and $r = \{r_1, \dots r_n\} \equiv r_1 \vee \cdots \vee r_n$ for $r_i \in \mathcal{B}_{RCCi}$. Then $l^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times (\Delta^*)^{\mathcal{I}}$; $r^{\mathcal{I}} = r_1^{\mathcal{I}} \cup \cdots \cup r_n^{\mathcal{I}}$; $(R \circ l)^{\mathcal{I}} = \{(d, e^*) \in \Delta^{\mathcal{I}} \times (\Delta^*)^{\mathcal{I}} \mid$ there is an e s.t. $(d, e) \in R^{\mathcal{I}}$ and $(e, e^*) \in l^{\mathcal{I}}\}$; $(\exists U_1, U_2.r)^{\mathcal{I}} = \{d \in \Delta^{\mathcal{I}} \mid$ there exist e_1^*, e_2^* s.t. $(d, e_1^*) \in U_1^{\mathcal{I}}, (d, e_2^*) \in U_2^{\mathcal{I}}$ and $(e_1^*, e_2^*) \in r^{\mathcal{I}}\}$; $(\forall U_1, U_2.r)^{\mathcal{I}} = \{d \in \Delta^{\mathcal{I}} \mid$ for all e_1^*, e_2^* s.t. $(d, e_1^*) \in U_1^{\mathcal{I}}, (d, e_2^*) \in U_2^{\mathcal{I}}$ it holds that $(e_1^*, e_2^*) \in r^{\mathcal{I}}\}$.

Now we can define the following combined geo-thematic logic (where a^*, b^* stand for constants intended to be interpreted by regions):

Definition 4 (DL-Lite$^{\sqcap}_{\mathcal{F},\mathcal{R}}$(RCC8)). *Let* $r \in Rel_{RCC8}$ *and* $T_\omega = Ax_{RCC8}$.

$$\begin{array}{ll}
R \longrightarrow P \mid P^- & U \longrightarrow R \mid \tilde{R} \quad B \longrightarrow A \mid \exists R \mid \exists l \\
C_l \longrightarrow B \mid C_l \sqcap B & C_r \longrightarrow B \mid \neg B \mid \exists U_1, U_2.r \\
\textit{TBox}^{*)}: & C_l \sqsubseteq C_r, (\text{funct } l), (\text{funct } R), R_1 \sqsubseteq R_2 \\
\textit{ABox}: & A(a), R(a,b), l(a, a^*), r(a^*, b^*)
\end{array}$$

**) Restriction: If* (funct R) $\in \mathcal{T}$, *then* R *and* R^- *do not occur on the right-hand side of a role inclusion axiom or in a concept of the form* $\exists U_1, U_2.r$.

As satisfiability checking of RCC8 constraint networks is NP-complete, there is only a chance to reach FOL rewritability if we assume within the ABox a constraint network which is consistent and complete, i.e., it has a exactly one solution and it is a clique with base relations as labels; in this case the ABox is called *spatially complete*. For cadastral maps or maps containing areas of administration one can assume pretty safely (almost) spatial completeness. The coupling with RCC8 is so weak that FOL rewritability of satisfiability follows.

Proposition 1. *Checking the satisfiability of* $DL-Lite^{\sqcap}_{\mathcal{F},\mathcal{R}}$*(RCC8) ontologies that have a spatially complete ABox is FOL rewritable.*

Testing whether FOL rewritability holds for satisfiability tests is necessary for tests whether FOL rewritability is provable for query answering w.r.t. a sufficiently expressive query language. The query language which we consider is derived from grounded conjunctive queries and is denoted by GCQ^+. This query language is explicitly constructed for use with DL-Lite$^{\sqcap}_{\mathcal{F},\mathcal{R}}$(RCC8) and so provides only means for qualitative spatial queries. But it could be extended to allow also for quantitative spatial queries.

Definition 5. *A* GCQ^+ atom w.r.t. DL-Lite$^{\sqcap}_{\mathcal{F},\mathcal{R}}$(RCC8) *is a formula of one of the following forms:*

- *$C(x)$, where C is a DL-Lite$^{\sqcap}_{\mathcal{F},\mathcal{R}}$(RCC8) concept without the negation symbol and x is a variable or a constant.*
- *$(\exists R_1 \dots R_n.C)(x)$ for role symbols or their inverses R_i, a DL-Lite$^{\sqcap}_{\mathcal{F},\mathcal{R}}$(RCC8) concept C without the negation symbol, and a variable or a constant x*
- *$R(x, y)$ for a role symbol R or an inverse thereof*
- *$l(x, y^*)$, where x is a variable or constant and y^* is a variable or constant intended to denote elements of Ax_{RCC8}*
- *$r(x^*, y^*)$, where $r \in Rel_{RCC8}$ and x^*, y^* are variables or constants intended to denote elements of Ax_{RCC8}*

A GCQ^+ query w.r.t. DL-Lite$^{\sqcap}_{\mathcal{F},\mathcal{R}}$(RCC8) *is a query $\tilde{\exists}\boldsymbol{y}\boldsymbol{z}^* \bigwedge C_i(\boldsymbol{x}, \boldsymbol{w}^*, \boldsymbol{y}, \boldsymbol{z}^*)$ where all $C_i(\boldsymbol{x}, \boldsymbol{w}^*, \boldsymbol{y}, \boldsymbol{z}^*)$ are GCQ^+ atoms and $\tilde{\exists}\boldsymbol{y}\boldsymbol{z}^* = \tilde{\exists}y_1 \dots \tilde{\exists}y_n \tilde{\exists}z_1^* \dots \tilde{\exists}z_m^*$ is a sequence of $\exists$-quantifiers interpreted w.r.t. the active domain semantics.*

Our perfect rewriting algorithm is an an adaptation of the algorithm PerfectRef [3, Fig. 13] for reformulating UCQs w.r.t. DL-Lite ontologies to our setting in which GCQ^+-queries are asked to DL-Lite$^{\sqcap}_{\mathcal{F},\mathcal{R}}$(RCC8) ontologies. We give a description of our adapted algorithm in the following.

Given a query GCQ^+ Q, we transform it to a special form; $\tau_1(Q)$ is the result of the transformation to a UCQ and $\tau_2(Q)$ is the result of transforming Q in a hybrid UCQ whose conjuncts are either classical predicate logical atoms or GCQ^+-atoms which are not further transformed. We use the notation "$g = F$" for "g is of the form F".

The original algorithm PerfectRef operates on the PI axioms of a DL-Lite ontology by using them as rewriting aids for the atomic formulas in the UCQ. Lines 5–12 and 28–34 of our adapted algorithm (Algorithm 1) make up the original PerfectRef. Roughly, the PerfectRef algorithm acts in the inverse direction with respect to the chasing process. For example, if the TBox contains the PI axiom $A_1 \sqcap A_2 \sqsubseteq A_3$, and the UCQ contains the atom $A_3(x)$ in a CQ, then the new rewritten UCQ query contains a CQ in which $A_3(x)$ is substituted by $A_1(x) \wedge A_2(x)$. The applicability of a PI axiom to an atom is restricted in those cases where the variables of an atom are either distinguished variables or also appear in another atom of the CQ at hand. To handle these cases, PerfectRef—as well as also our adapted version—uses anonymous variables $\_$ to denote all non-distinguished variables in an atom that do not occur in other atoms of the same CQ. The function anon (line 31 in Algorithm 1) implements the anonymization. The application conditions for PI axioms α and atoms are as follows: α is applicable to $A(x)$ if A occurs on the right-hand side; and α is applicable to $R(x_1, x_2)$, if $x_2 = \_$ and the right-hand side of α is $\exists R$; or $x_1 = \_$ and the right-hand side of α is $\exists R^-$; or α is a role inclusion assertion and its right-hand side is either R or R^-. The outcome $gr(g, \alpha)$ of applying an applicable PI α to an atom g corresponds to the outcome of resolving α with g. For example, if α is $A \sqsubseteq \exists R$ and g is $R(x, \_)$, the result of the application is $gr(g, \alpha) = A(x)$. We leave out the details [3, Fig.12, p. 307]. In PerfectRef, atoms in a CQ are rewritten with the PI axioms (lines 6–11) and if possible merged by the function reduce (line 31) which unifies the atoms with the most general unifier (lines 28–34).

```
   input  : a hybrid query $\tau_1(Q) \cup \tau_2(Q)$, DL-Lite(RCC8) TBox $\mathcal{T}$
   output: a UCQ $pr$
1  $pr := \tau_1(Q) \cup \tau_2(Q)$;
2  repeat
3      $pr' := pr$;
4      foreach $query\ q' \in pr'$ do
5          foreach $atom\ g\ in\ q'$ do
6              if $g$ is a FOL-atom then
7                  foreach $PI\ \alpha\ in\ \mathcal{T}$ do
8                      if $\alpha$ is applicable to $g$ then
9                          $pr := pr \cup \{q'[g/gr(g, \alpha)]\}$;
10                     end
11                 end
12             else
13                 if $g = \exists \tilde{R}_1, \tilde{R}_2.r_3(x)$ and $r_1; r_2 \subseteq r_3$ then
14                     $X := q'[g/(\exists \tilde{R}_1, l.r_1(x) \wedge \exists l, \tilde{R}_2.r_2(x))]$;
15                     $pr := pr \cup \{X\} \cup \{\tau_2\big(X, \{\exists \tilde{R}_1, l.r_1(x), \exists l, \tilde{R}_2.r_2(x)\}\big)\}$
16                 end
17                 if $g = \exists U_1, U_2.r_1(x)$ and $B \sqsubseteq \exists U_1, U_2.r_2(x) \in \mathcal{T}$ for $r_2 \subseteq r_1$
                   then
18                     $pr := pr \cup \{q'[g/B(x)]\}$;
19                 end
20                 if $g = \exists U_1, U_2.r_1(x)$ and $B \sqsubseteq \exists U_1, U_2.r_2(x) \in \mathcal{T}$ for $r_2^{-1} \subseteq r_1$
                   then
21                     $pr := pr \cup \{q'[g/B(x)]\}$;
22                 end
23                 if $g = \exists \tilde{R}_1, U_1.r(x)$ (resp. $\exists U_1, \tilde{R}_1.r(x)$) and ($R_2 \sqsubseteq R_1 \in \mathcal{T}$ or
                   $R_2^{-1} \sqsubseteq R_1^{-1} \in \mathcal{T}$) then
24                     $X := q'[g/(g[R_1/R_2])]$;
25                     $pr := pr \cup \{X\} \cup \{\tau_2\big(X, \{g[R_1/R_2]\}\big)\}$;
26                 end
27             end
28         end
29         foreach pair of FOL-atoms $g_1, g_2$ in $q'$ do
30             if $g_1$ and $g_2$ unify then
31                 $pr := pr \cup \{anon(reduce(q', g_1, g_2))\}$;
32             end
33         end
34     end
35 until $pr' = pr$;
36 return $drop(pr)$
```

Algorithm 1. Adapted PerfectRef

The modification of PerfectRef concerns the handling of GCQ^+-atoms of the form $\exists U_1, U_2.r(x)$. These atoms may have additional implications that are accounted for with four cases (lines 12–26 of the algorithm). At the end of the adapted algorithm PerfectRef (Algorithm 1, line 35) these atoms are deleted by calling the function *drop*. The algorithm returns a classical UCQ, which can be evaluated as a SQL query on the database $DB(\mathcal{A})$.

Let us demonstrate the rewriting algorithm with a formalization of the simple scenario from the beginning. The TBox of the engineering bureau contains the following axioms of DL-Lite$^{\sqcap}_{\mathcal{F},\mathcal{R}}$(RCC8): $\mathit{Park{+}Lake} \sqsubseteq \mathit{Park}$; $\mathit{Park4Playing} \sqsubseteq \mathit{Park}$; $\mathit{Park{+}Lake} \sqsubseteq \exists \mathit{hasLake} \circ l, l.\mathsf{tpp}$; $\mathit{Park4Playing} \sqsubseteq \exists \mathit{hasPlAr} \circ l, l.\mathsf{tpp}$. The ABox $\mathcal{A}$ contains $\{\mathit{Park{+}Lake}(a), \mathit{Park4Playing}(a)\} \subseteq \mathcal{A}$. The query of the engineer, which asks for all parks with lakes and playing areas such that the playing area is not a tangential proper part of the lake, can be formalized by the following GCQ^+: $Q = \mathit{Park}(x) \wedge \exists \mathit{hasLake} \circ l, \mathit{hasPlAr} \circ l.(\mathcal{B}_{RCC8} \setminus \{\mathsf{ntpp}\})(x)$. Using the composition $\mathsf{tpp};\mathsf{tppi} = \{\mathsf{dc}, \mathsf{ec}, \mathsf{po}, \mathsf{tpp}, \mathsf{tppi}, \mathsf{eq}\} \subseteq \mathcal{B}_{RCC8} \setminus \{\mathsf{ntpp}\}$, the reformulation algorithm introduced above (lines 13–15) produces a UCQ that contains the following CQ: $Q' = (\exists \mathit{hasLake} \circ l, l.\mathsf{tpp})(x) \wedge (\exists l, \mathit{hasPlAr} \circ l.\mathsf{tppi})(x)$. Rewriting $\exists l, \mathit{hasPlAr} \circ l.\mathsf{tppi}$ to $\exists \mathit{hasPlAr} \circ l, l.\mathsf{tpp}$ (lines 20–21) in combination with the rewriting rule for $A_1 \sqsubseteq A_2$ (Def. 3) we get another CQ $Q'' = \mathit{Park{+}Lake}(x) \wedge \mathit{Park4Playing}(x)$. Now, Q'' captures (as desired) the object a.

That the rewriting given in Algorithm 1 is indeed correct and complete follows from Theorem 1.

Theorem 1. *Answering GCQ^+-queries w.r.t. $DL - Lite^{\sqcap}_{\mathcal{F},\mathcal{R}}$(RCC8) ontologies that have a spatially complete ABox is FOL-rewritable.*

We give a proof sketch. The proof follows the proof of Theorem 5.15 for pure DL-Lite ontologies [3]. We adapt the chase construction to account for the RCC8 relations $r \in Rel_{RCC8}$. The main observation is that the disjunctions in r can be nearly handled as if they were predicate symbols.

Because of Prop. 1 we may assume that $\mathcal{O}$ is satisfiable. Let *pr* be the UCQ resulting from applying the algorithm to Q and $\mathcal{O}$. We have to show that $cert(Q, \mathcal{O}) = (pr)^{DB(\mathcal{A})}$. These can be done in two main steps of which the first will be sketched here as it contains the main chase-like construction $chase^*(\mathcal{O})$. After the construction, one has to describe what it means to answer Q with respect to $chase^*(\mathcal{O})$ is, resulting in the set $ans(Q, chase^*(\mathcal{O}))$, and then show that $ans(Q, chase^*(\mathcal{O})) = cert(Q, \mathcal{O})$. In the second step, which we leave out here (see the extended version of this paper) one has to show that $ans(Q, chase^*(\mathcal{O})) = (pr)^{DB(\mathcal{A})}$.

For the construction of $chase^*(\mathcal{O})$ one uses the chase rules of Def. 3 and the special rule (R).

Chasing Rule (R)

If $B(x) \in S_i$ and there are no y, y^*, x^* s.t. $\{R_1(x, y), l(y, y^*), l(x, x^*), r_1(y^*, x^*)\}$ is contained in S_i, then let $S_{i+1} = S_i \cup \{R_1(x, y), l(y, y^*), l(x, x^*), r_1(y^*, x^*)\}$. The constants y, y^* are completely new constants not appearing in S_i. The constant x^* is the old x^* if already in S_i, otherwise it is also a completely new constant symbol.

Every time (R) is applied to yield a new ABox S_i, the resulting constraint network in S_i is saturated by calculating the minimal labels between the new added region constants and the other region constants. The application of (R) does not constrain the RCC8-relations between the old regions and even stronger: Let (R) be applied to a TBox axiom of the form $A \sqsubseteq \exists \tilde{R}, l.r$ and $A(a) \in S_i$ resulting in the addition of $R(a, b)$, $l(b, b^*)$ and $r(b^*, a^*)$. Then it is enough to consider all $c^* \in S_i$ and all relations r_{c^*,a^*} such that $r_{c^*,a^*}(c^*, a^*) \in S_i$. The composition table gives the outcome $r_{c^*,a^*}; r = r'_{c^*,b^*}$ and one adds $r'_{c^*,b^*}(c^*, b^*)$ to S_i. After this step, which we call triangulation step, one closes the assertions up with respect to the subset relation between RCC8-relations and with respect to symmetry. I.e., if $r_1(x^*, y^*)$ is added to S_i, then one also adds $r_2(x^*, y^*)$ for all r_2 such that $r_1 \subseteq r_2$ and $r_2^{-1}(y^*, x^*)$. For different c_1^*, c_2^*, assertions of the form $r_{c_1^*,b^*}(c_1^*, b^*)$ and $r_{c_2^*,b^*}(c_2^*, b^*)$ do not constrain each other (because of the patch work property). The saturation leads to a finite set S_{i+k} (for some $k \in \mathbb{N}$) that is a superset of S_i. Let $chase^*(\mathcal{O}) = \bigcup S_i$. The set $chase^*(\mathcal{O})$ does not induce a single canonical model. But it is universal in the following sense: For every model $\mathcal{I}$ of $\mathcal{O}$ define a model $\mathcal{I}_c$ out of $chase^*(\mathcal{O})$ by taking a (consistent) configuration of the contained RCC8-network and taking the minimal model of this configuration and the thematic part of $chase^*(\mathcal{O})$. Then $\mathcal{I}_c$ maps homomorphically to $\mathcal{I}$. Now one can define that answers of GCQ^+-queries with respect to $chase^*(\mathcal{O})$ are given by homomorphic embeddings and show that these answers are exactly the certain answers w.r.t. the ontology $\mathcal{O}$.

4 Strong Combinations of DL-Lite with RCC

Another way of reaching FOL rewritability for combinations of DL-Lite with RCC is weakening the expressivity of the spatial component. Hence, one may ask whether a combination with the calculus RCC3 or RCC2 [17], both fragments with weak expressibility, allows for weak FOL rewritability w.r.t. satisfiability checks (and query answering). Their potential use as logics for approximating [6] ontologies in more expressible combined logics like $\mathcal{ALC}$(RCC8) makes the investigation valuable. The logics DL-Lite$^{\sqcap,+}_{\mathcal{F},\mathcal{R}}$(RCC2) and DL-Lite$^{\sqcap,+}_{\mathcal{F},\mathcal{R}}$(RCC3) are defined as follows ('+' indicates the strong combination):

Definition 6 (DL-Lite$^{\sqcap,+}_{\mathcal{F},\mathcal{R}}$(RCC2) and DL-Lite$^{\sqcap,+}_{\mathcal{F},\mathcal{R}}$(RCC3)). *Let $T_\omega = Ax_{RCC2}$ resp. $T_\omega = Ax_{RCC3}$ and $r \in \mathcal{B}_{RCC2}$ resp. $r \in \mathcal{B}_{RCC3}$*

$$\begin{array}{llll}
R \longrightarrow P \mid P^- & U \longrightarrow l \mid \tilde{R} & B \longrightarrow A \mid \exists R \\
C_l \longrightarrow B \mid C_l \sqcap B & C_r \longrightarrow B \mid \neg B \mid \exists U_1, U_2.r & \\
\textit{TBox}^{*)}: & C_l \sqsubseteq C_r, (\text{funct } l, R), R_1 \sqsubseteq R_2 & \\
\textit{ABox}: & A(a), R(a,b), l(a,a^*), r(a^*,b^*) &
\end{array}$$

**) Restriction: If* (funct R) $\in \mathcal{T}$*, then R and R^- do not occur on the right-hand side of a role inclusion axiom.*

For RCC3 the strong combination with DL-Lite$^{\sqcap}_{\mathcal{F},\mathcal{R}}$ leads to non-FOL rewritability. The reason lies in the fact that testing the satisfiability of RCC3 is not in the complexity class AC^0 as shown by the following lemma.

Lemma 1. *Checking satisfiability of RCC3 networks is* LOGSPACE *hard.*

Proof. As is known, the reachability problem in symmetric (undirected) graphs is logspace complete [12]—where graph reachability asks whether for nodes s, t in G there is a path between s and t. By reducing this problem to the satisfiability test for RCC3 networks we will have shown that the latter problem is LOGSPACE hard itself. So let be given a (symmetric) graph $G = (V, E)$ and nodes $s, t \in V$. We define the network N in the following way (see Figure 4): Let $V = \{v_1, \dots, v_n\}$ be an enumeration of the nodes of G; w.l.o.g. let $s = v_1$ and $t = v_n$ and let $\mathcal{B} = \mathcal{B}_{RCC3}$. Nodes of N are given by $V \cup \{a\}$ where $a \notin V$. Labelled edges of N are given by: $s\{\mathsf{DR}\}a$; $t\{\mathsf{ONE}\}a$; $v_i\{\mathcal{B}\}a$ for all $i \neq 1, n$; $v_i\{\mathsf{EQ}\}v_j$ if $E(v_i, v_j)$; $v_i\{\mathcal{B}\}v_j$ if $\neg E(v_i, v_j)$.

Now we show that the network N is satisfiable iff s and t are connected in G. Assume that s and t are connected; then there is an EQ-path in N between them, hence $s\{\mathsf{EQ}\}t$ follows. But this contradicts $s\{\mathsf{DR}\}a$ and $t\{\mathsf{ONE}\}a$. Now assume that s and t are not connected; then there is no path consisting only of EQ-labels between s and t. The graph G consists of at least 2 components, and s, t are in different components. We define a consistent configuration as follows: For all nodes v, v' in the component in which s is contained, let $v\{\mathsf{DR}\}a$ and $v\{\mathsf{EQ}\}v'$. For all nodes v, v' in the component of t let $v\{\mathsf{ONE}\}a$ and $v\{\mathsf{EQ}\}v'$. For all nodes v, v' in the other components let $v\{\mathsf{DR}\}a$ and $v\{\mathsf{EQ}\}v'$. For all nodes v, v' which have not a label yet, let $v\{\mathsf{DR}\}v'$. (Two remarks : 1) EQ-edges for edges $E(v_i, v_j)$ in G with $j > i + 1$ are not shown in Fig. 4. 2) We inserted edges labelled $\mathcal{B}$ for better illustrations. But these are not needed.)

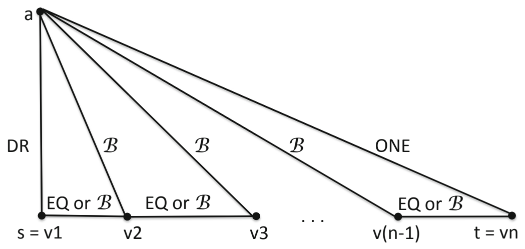


Fig. 4. Network N used in proof of Lemma 1

This lemma immediately entails the fact that satisfiability checking for ontologies over the logic DL-Lite$^{\sqcap,+}_{\mathcal{F},\mathcal{R}}$(RCC3) is not FOL rewritable. This problem does not vanish if we presuppose that the ABox $\mathcal{A}$ is spatially complete—as shown by the following proposition.

Proposition 2. *Satisfiability checking of ontologies in DL-Lite$^{\sqcap,+}_{\mathcal{F},\mathcal{R}}$(RCC3) with spatially complete ABoxes is not FOL rewritable.*

Proof. We construct a generic TBox $\mathcal{T}_g$ that allows one to encode any RCC3 constraint network so that checking the consistency of RCC3 constraint networks is reducible to a satisfiability check of this TBox and a spatially complete ABox. Let for every $r \in Rel_{RCC3}$ be given role symbols R_r^1, R_r^2. The generic TBox $\mathcal{T}_g$ has for every $r \in Rel_{RCC3}$ a concept symbol A_r and a corresponding axiom with the content that all instances of A_r have paths over the abstract features R_1 resp. R_2 to regions that are r-related.

$$\mathcal{T}_g = \{A_r \sqsubseteq \exists \tilde{R}_r^1, \tilde{R}_r^2.r, (\text{funct } l, R_r^1, R_r^2) \mid r \in Rel_{RCC3}\} \tag{1}$$

Now, let N be an arbitrary RCC3 constraint network which has to be tested for relational consistency. Let $\mathcal{A}_N$ be an ABox such that for every $r(a,b)$ in N three new constants are introduced: x_{ab}, x_a, x_b.

$$\mathcal{A}_N = \{A_r(x_{ab}), R_r^1(x_{ab}, x_a), R_r^2(x_{ab}, x_b) \mid r(a,b) \in N\} \tag{2}$$

The construction entails: $\mathcal{T}_g \cup \mathcal{A}_N \cup Ax_{RCC3}$ is satisfiable iff $N \cup Ax_{RCC3}$ is satisfiable. If the data complexity of the satisfiability check for DL-Lite$^{\sqcap,+}_{\mathcal{F},\mathcal{R}}$(RCC3)-ontologies were in AC^0, then the consistency of constraint networks could be tested in AC^0, too. (Note that $\mathcal{T}_g$ is a fixed TBox.) But checking the consistency of RCC3 constraint networks is Logspace-hard and AC$^0 \subsetneq$ Logspace.

As a corollary to this proposition we get the assertion that strong combinations of RCC5 and RCC8 into DL-Lite$^{\sqcap,+}_{\mathcal{F},\mathcal{R}}$(RCC5) and DL-Lite$^{\sqcap,+}_{\mathcal{F},\mathcal{R}}$(RCC8), respectively—which are defined in the same manner as in Definition 6—do not allow for FOL rewritability of satisfiability checking.

The low resolution calculus RCC2 is quite more inexpressive than RCC3 due to the fact that the composition table does not allow for the propagation of information: All compositions of DR, O result in the maximally unspecified relation $\{\mathsf{DR}, \mathsf{O}\}$. Hence, FOL rewritability of satisfiability testing follows easily considering the query $Q = \exists x, y[\mathsf{O}(x,y) \wedge \mathsf{DR}(x,y)] \vee \exists x[\mathsf{DR}(x,x)]$.

Proposition 3. *Testing the satisfiability of RCC2 networks is FOL rewritable.*

But in combination with functionality axioms of the TBox one could have the problem that the ABox may lead to identifications of regions. The identified regions are not allowed to have edges labelled O, DR resp. to the same region. Though this can be tested, the problem arises when a chain of regions is identified by the TBox and the ABox, because we do not know the length of the chain in advance. More formally: In addition to RCC2 constraint-network assertions we allow identity assertions $v = w$ for regions v, w. As we can assume that all nodes in a RCC2 network are connected by an edge labelled O, DR or $\mathcal{B}_{RCC2}$ we use a more intuitive formalism where, for every assertion $v = w$, the label of the edge between v and w is marked with an $=$; e.g., an edge between v, w with label $\mathsf{DR}^=$ stands for $\mathsf{DR}(v,w) \wedge v = w$. We call such a network an $=$-marked RCC2 network (a RCC$^=$2 network for short). Let $\mathcal{B} = \mathcal{B}_{RCC2}$ in the following.

Proposition 4. *An RCC$^{=}$2 constraint network N is unsatisfiable iff*

1. *N contains $\mathsf{DR}(v,v)$ or $\mathsf{DR}^{=}(v,v)$ for some node v; or*
2. *N contains $\mathsf{DR}^{=}(v,w)$; or*
3. *N contains a cycle in which there is $\mathsf{DR}(v,w)$ and in which there is a path from v to w such that every label on the path is $\mathcal{B}^{=}$ or $\mathsf{O}^{=}$; or*
4. *N contains a cycle in which there is $\mathsf{DR}(v,w)$ and in which there is a path from v to w s.t. every label on the path is $\mathcal{B}^{=}$ or $\mathsf{O}^{=}$ except one which is O.*

Proposition 4 shows that adding identity assertions to an RCC2 network may require checking the existence of identity chains of arbitrary length. Hence, in principle it is possible that the functional roles used in DL-Lite$^{\sqcap,+}_{\mathcal{F},\mathcal{R}}$(RCC2) may lead to identity chains. But as the following proposition show, this cannot be the case: The identity paths induced by functionalities in DL-Lite$^{\sqcap,+}_{\mathcal{F},\mathcal{R}}$(RCC2) can have only a maximal length of one.

Proposition 5. *Satisfiability checking of ontologies in DL-Lite$^{\sqcap,+}_{\mathcal{F},\mathcal{R}}$(RCC2) is FOL rewritable.*

5 Conclusion

As proved in this paper, combining DL-Lite with expressive fragments of the region calculus like RCC8 into logics that preserve the property of FOL rewritability is possible if the coupling is weak: Constraints of the RCC8 network contained in the ABox are not transported over to the implicitly constructed constraint network resulting from the constructors of the form $\exists U_1, U_2.r$. In this paper we further dealt with strong combinations for weaker calculi like RCC2 or RCC3. As we have shown by a reduction proof, a strong combination with RCC3 destroys the FOL rewritability of satisfiability checking. The reason is that checking the satisfiability of RCC3 networks needs to test for reachability along EQ paths, which can be reproduced by the TBox. For the low resolution calculus RCC2, FOL rewritability of satisfiability checking is provable—though checking the satisfiability of RCC2 networks with additional identity assertions is at least as hard as checking RCC3 networks. We plan to investigate whether DL-Lite$^{\sqcap,+}_{\mathcal{F},\mathcal{R}}$(RCC2) and DL-Lite$^{\sqcap}_{\mathcal{F},\mathcal{R}}$(RCC8) can be used for approximation—following the complete but not necessarily correct approximation method of [6]. Moreover we want to check whether DL-Lite$^{\sqcap,+}_{\mathcal{F},\mathcal{R}}$(RCC2) allows for FOL rewritability of query answering w.r.t. unions of conjunctive queries.

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. Calì, A., Gottlob, G., Lukasiewicz, T.: A general datalog-based framework for tractable query answering over ontologies. Technical Report CL-RR-10-21, Oxford University Computing Laboratory (November 2010)

3. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Poggi, A., Rodriguez-Muro, M., Rosati, R.: Ontologies and Databases: The *DL-Lite* Approach. In: Tessaris, S., Franconi, E., Eiter, T., Gutierrez, C., Handschuh, S., Rousset, M.-C., Schmidt, R.A. (eds.) Reasoning Web 2009. LNCS, vol. 5689, pp. 255–356. Springer, Heidelberg (2009)
4. Grigni, M., Papadias, D., Papadimitriou, C.H.: Topological inference. In: IJCAI (1), pp. 901–907 (1995)
5. Haarslev, V., Lutz, C., Möller, R.: A description logic with concrete domains and a role-forming predicate operator. J. Log. Comput. 9(3), 351–384 (1999)
6. 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)
7. Kuper, G.M., Libkin, L., Paredaens, J. (eds.): Constraint Databases. Springer (2000)
8. Lutz, C., Miličić, M.: A tableau algorithm for description logics with concrete domains and general TBoxes. J. Autom. Reasoning 38(1-3), 227–259 (2007)
9. Özçep, Ö.L., Möller, R.: Combining DL-Lite with spatial calculi for feasible geo-thematic query answering. In: Kazakov, Y., Lembo, D., Wolter, F. (eds.) Proceedings of the 25th Iternational Workshop on Description Logics (DL 2012), vol. 846 (2012), http://ceur-ws.org/Vol-846/
10. Özçep, Ö.L., Möller, R.: Computationally feasible query answering over spatio-thematic ontologies. In: Proceedings of GEOProcessing 2012, The Fourth International Conference on Advanced Geographic Information Systems, Applications, and Services (2012), http://www.thinkmind.org/index.php?view=article&articleid=geoprocessing_2012_7_10_30059
11. Randell, D.A., Cui, Z., Cohn, A.G.: A spatial logic based on regions and connection. In: Proceedings of the 3rd International Conferecence on Knowledge Representation and Reasoning, pp. 165–176 (1992)
12. Reingold, O.: Undirected connectivity in log-space. J. ACM 55(4), 17:1–17:24 (2008), http://doi.acm.org/10.1145/1391289.1391291
13. Renz, J.: Qualitative Spatial Reasoning with Topological Information. LNCS (LNAI), vol. 2293. Springer, Heidelberg (2002)
14. Renz, J., Nebel, B.: Qualitative spatial reasoning using constraint calculi. In: Aiello, M., Pratt-Hartmann, I., Benthem, J. (eds.) Handbook of Spatial Logics, pp. 161–215. Springer, Netherlands (2007)
15. Rodriguez-Muro, M., Calvanese, D.: Semantic index: Scalable query answering without forward chaining or exponential rewritings. Posters of the 10th Int. Semantic Web Conf., ISWC 2011 (2011)
16. Wessel, M.: On spatial reasoning with description logics - position paper. In: Ian Horrocks, S.T. (ed.) Proceedings of the International Workshop on Description Logics (DL 2002). CEUR Workshop Proceedings, vol. 53 (2002), http://CEUR-WS.org/Vol-53/
17. Wessel, M.: Qualitative spatial reasoning with the $\mathcal{ALCI}_{RCC}$- family — first results and unanswered questions. Technical Report FBI–HH–M–324/03, University of Hamburg, Department for Informatics (2003)

Author Index