Grigoris Antoniou Marko Grobelnik
Elena Simperl Bijan Parsia
Dimitris Plexousakis Pieter De Leenheer
Jeff Pan (Eds.)

LNCS 6643

The Semantic Web: Research and Applications

8th Extended Semantic Web Conference, ESWC 2011
Heraklion, Crete, Greece, May/June 2011
Proceedings, Part I

1 Part I

Springer

Lecture Notes in Computer Science 6643

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

Editorial Board

David Hutchison
Lancaster University, UK
Takeo Kanade
Carnegie Mellon University, Pittsburgh, PA, USA
Josef Kittler
University of Surrey, Guildford, UK
Jon M. Kleinberg
Cornell University, Ithaca, NY, USA
Alfred Kobsa
University of California, Irvine, CA, USA
Friedemann Mattern
ETH Zurich, Switzerland
John C. Mitchell
Stanford University, CA, USA
Moni Naor
Weizmann Institute of Science, Rehovot, Israel
Oscar Nierstrasz
University of Bern, Switzerland
C. Pandu Rangan
Indian Institute of Technology, Madras, India
Bernhard Steffen
TU Dortmund University, Germany
Madhu Sudan
Microsoft Research, Cambridge, MA, USA
Demetri Terzopoulos
University of California, Los Angeles, CA, USA
Doug Tygar
University of California, Berkeley, CA, USA
Gerhard Weikum
Max Planck Institute for Informatics, Saarbruecken, Germany

Grigoris Antoniou Marko Grobelnik
Elena Simperl Bijan Parsia
Dimitris Plexousakis Pieter De Leenheer
Jeff Pan (Eds.)

The Semantic Web: Research and Applications

8th Extended Semantic Web Conference, ESWC 2011
Heraklion, Crete, Greece, May 29 – June 2, 2011
Proceedings, Part I

Springer

Volume Editors

Grigoris Antoniou
FORTH-ICS and University of Crete, 71110 Heraklion, Crete, Greece
E-mail: antoniou@ics.forth.gr

Marko Grobelnik
Jožef Stefan Institute, Jamova 39, 1000 Ljubljana, Slovenia
E-mail: marko.grobelnik@ijs.si

Elena Simperl
Karlsruhe Institute of Technology, 76128 Karlsruhe, Germany
E-mail: elena.simperl@aifb.uni-karlsruhe.de

Bijan Parsia
University of Manchester, Manchester M13 9PL, UK
E-mail: bparsia@cs.man.ac.uk

Dimitris Plexousakis
FORTH-ICS and University of Crete, 70013 Heraklion, Crete, Greece
E-mail: dp@ics.forth.gr

Pieter De Leenheer
VU University of Amsterdam, 1012 ZA Amsterdam, The Netherlands
E-mail: pgm.de.leenheer@few.vu.nl

Jeff Pan
University of Aberdeen, Aberdeen AB24 3UE, UK
E-mail: jeff.z.pan@abdn.ac.uk

ISSN 0302-9743 e-ISSN 1611-3349
ISBN 978-3-642-21033-4 e-ISBN 978-3-642-21034-1
DOI 10.1007/978-3-642-21034-1

Springer Heidelberg Dordrecht London New York

Library of Congress Control Number: 2011926990

CR Subject Classification (1998): H.4, H.3.3, H.5, J.4, I.2.4, K.4.2

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

© Springer-Verlag Berlin Heidelberg 2011

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, re-use of illustrations, recitation, broadcasting, reproduction on microfilms or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer. Violations are liable to prosecution under the German Copyright Law.
The use of general descriptive names, registered names, trademarks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

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

Every year ESWC brings together researchers and practitioners dealing with different aspects of semantic technologies. Following a successful re-launch in 2010 as a multi-track conference, the 8th Extended Semantic Web Conference built on the success of the ESWC conference series initiated in 2004. Through its extended concept this series seeks to reach out to other communities and research areas, in which Web semantics play an important role, within and outside ICT, and in a truly international, not just 'European' context. This volume contains the papers accepted for publication in key tracks of ESWC 2011: the technical tracks including research tracks, an in-use track and two special tracks, as well as the PhD symposium and the demo track.

Semantic technologies provide machine-understandable representations of data, processes and resources — hardware, software and network infrastructure — as a foundation for the provisioning of a fundamentally new level of functionality of IT systems across application scenarios and industry sectors. Using automated reasoning services over ontologies and metadata, semantically enabled systems will be able to better interpret and process the information needs of their users, and to interact with other systems in an interoperable way. Research on semantic technologies can benefit from ideas and cross-fertilization with many other areas, including artificial intelligence, natural language processing, database and information systems, information retrieval, multimedia, distributed systems, social networks, Web engineering, and Web science. These complementarities are reflected in the outline of the technical program of ESWC 2011; in addition to the research and in-use tracks, we furthermore introduced two special tracks this year, putting particular emphasis on inter-disciplinary research topics and areas that show the potential of exciting synergies for the future. In 2011, these special tracks focused on data-driven, inductive and probabilistic approaches to managing content, and on digital libraries, respectively.

The technical program of the conference received 247 submissions, which were reviewed by the Program Committee of the corresponding tracks. Each track was coordinated by Track Chairs and installed a dedicated Program Committee. The review process included paper bidding, assessment by at least three Program Committee members, and meta-reviewing for each of the submissions that were subject to acceptance in the conference program and proceedings. In all, 57 papers were selected as a result of this process, following comparable evaluation criteria devised for all technical tracks.

The PhD symposium received 25 submissions, which were reviewed by the PhD Symposium Program Committee. Seven papers were selected for presentation at a separate track and for inclusion in the ESWC 2011 proceedings. The demo track received 19 submissions, 14 of which were accepted for demonstration in a dedicated session during the conference. Ten of the demo papers were also selected for inclusion in the conference proceedings.

ESWC 2011 had the pleasure and honor to welcome seven renowned keynote speakers from academia and industry, addressing a variety of exciting topics of highest relevance for the research agenda of the semantic technologies community and its impact on ICT:

– James Hendler, Tetherless World Professor of Computer and Cognitive Science and Assistant Dean for Information Technology and Web Science at Rensselaer Polytechnic Institute
– Abe Hsuan, founding partner of the law firm Irwin & Hsuan LLP
– Prasad Kantamneni, principal architect of the Eye-Tracking platform at Yahoo!
– Andraž Tori, CTO and co-founder of Zemanta
– Lars Backstrom, data scientist at Facebook
– Jure Leskovec, assistant professor of Computer Science at Stanford University
– Chris Welty, Research Scientist at the IBM T.J. Watson Research Center in New York

We would like to take the opportunity to express our gratitude to the Chairs, Program Committee members and additional reviewers of all refereed tracks, who ensured that ESWC 2011 maintained its highest standards of scientific quality. Our thanks should also reach the Organizing Committee of the conference, for their dedication and hard work in selecting and coordinating the organization of a wide array of interesting workshops, tutorials, posters and panels that completed the program of the conference. Special thanks go to the various organizations who kindly support this year's edition of the ESWC as sponsors, to the Sponsorship Chair who coordinated these activities, and to the team around STI International who provided an excellent service in all administrative and logistic issues related to the organization of the event. Last, but not least, we would like to say thank you to the Proceedings Chair, to the development team of the Easychair conference management system and to our publisher, Springer, for their support in the preparation of this volume and the publication of the proceedings.

May 2011

Grigoris Antoniou
Marko Grobelnik
Elena Simperl
Bijan Parsia
Dimitris Plexousakis
Pieter de Leenheer
Jeff Pan

Organization

Organizing Committee

General Chair	Grigoris Antoniou (FORTH-ICS and University of Crete, Greece)
Program Chairs	Marko Grobelnik (Jozef Stefan Institute, Slovenia)
	Elena Simperl (Karlsruhe Institute of Technology, Germany)
News from Front Coordinators	Lyndon Nixon (STI International, Austria)
	Alexander Wahler (STI International, Austria)
Poster and Demo Chairs	Bijan Parsia (University of Manchester, UK)
	Dimitris Plexousakis (FORTH-ICS and University of Crete, Greece)
Workshop Chairs	Dieter Fensel (University of Innsbruck, Austria)
	Raúl García Castro (UPM, Spain)
Tutorials Chair	Manolis Koubarakis (University of Athens, Greece)
PhD Symposium Chairs	Jeff Pan (University of Aberdeen, UK)
	Pieter De Leenheer (VU Amsterdam, The Netherlands)
Semantic Technologies Coordinators	Matthew Rowe (The Open University, UK)
	Sofia Angelatou (The Open University, UK)
Proceedings Chair	Antonis Bikakis (University of Luxembourg, Luxembourg)
Sponsorship Chair	Anna Fensel (FTW, Austria)
Publicity Chair	Lejla Ibralic-Halilovic (STI, Austria)
Panel Chairs	John Domingue (The Open University, UK)
	Asuncion Gomez-Perez (UPM, Spain)
Treasurer	Alexander Wahler (STI International, Austria)
Local Organization and Conference Administration	STI International, Austria

Program Committee - Digital Libraries Track

Track Chairs

Martin Doerr, Carlo Meghini and Allen Renear

Members

Trond Aalberg	Dimitris Kotzinos
Bruno Bachimont	Marcia Leizeng
Donatella Castelli	Eva Méndez
Panos Constantopoulos	Alistair Miles
Stefan Gradmann	John Mylopoulos
Jen-Shin Hong	Carole Palmer
Eero Hyvönen	Ingeborg Torvik Solvberg
Antoine Isaac	Douglas Tudhope
Traugott Koch	Herbert Van De Sompel

Program Committee - Inductive and Probabilistic Approaches Track

Track Chairs

Rayid Ghani and Agnieszka Lawrynowicz

Members

Sarabjot Anand	Ross King
Mikhail Bilenko	Jens Lehmann
Stephan Bloehdorn	Yan Liu
Chad Cumby	Matthias Nickles
Claudia D'Amato	Sebastian Rudolph
Nicola Fanizzi	Dou Shen
Blaz Fortuna	Sergej Sizov
Eric Gaussier	Umberto Straccia
Melanie Hilario	Vojtech Svatek
Luigi Iannone	Volker Tresp
Ivan Jelinek	Joaquin Vanschoren
Jörg-Uwe Kietz	

Program Committee - Linked Open Data Track

Track Chairs

Mariano Consens, Paul Groth and Jens Lehmann

Members

José Luis Ambite	Lin Clark
Sören Auer	Richard Cyganiak
Christian Bizer	Christophe Gueret
Paolo Bouquet	Harry Halpin
Dan Brickley	Andreas Harth

Olaf Hartig
Oktie Hassanzadeh
Sebastian Hellmann
Rinke Hoekstra
Anja Jentzsch
Spyros Kotoulas
Yuzhong Qu
Yves Raimond
Juan F. Sequeda
Nigam Shah
York Sure
Thanh Tran
Mischa Tuffield
Jun Zhao

Program Committee - Mobile Web Track

Track Chairs

Ora Lassila and Alessandra Toninelli

Members

Paolo Bellavista
Cristian Borcea
Valerie Issarny
Deepali Khushraj
Mika Mannermaa
Enrico Motta
Massimo Paolucci
Terry Payne
David Provost
Anand Ranganathan
Bernhard Schandl
Matthias Wagner

Program Committee - Natural Language Processing Track

Track Chairs

Philipp Cimiano and Michael Witbrock

Members

Guadalupe Aguado-De-Cea
Enrique Alfonseca
Nathalie Aussenac Gilles
Roberto Basili
Kalina Bontcheva
Christopher Brewster
Peter Clark
Thierry Declerck
Blaz Fortuna
Aldo Gangemi
Claudio Giuliano
Gregory Grefenstette
Siegfried Handschuh
Laura Hollink
Andreas Hotho
José Iria
Vanessa Lopez
Diana Maynard
John Mccrae
Paola Monachesi
Roberto Navigli
Achim Rettinger
Marta Sabou
Sergej Sizov
Pavel Smrz
Dennis Spohr
Christina Unger
Victoria Uren
Johanna V"olker
René Witte
Fabio Massimo Zanzotto

Program Committee - Ontologies Track

Track Chairs

Mathieu D'Aquin and Heiner Stuckenschmidt

Members

Nathalie Aussenac-Gilles
Eva Blomqvist
Ales Buh
Jean Charlet
Oscar Corcho
Isabel Cruz
Jesualdo Tomás Fernández-Breis
Chiara Ghidini
Asun Gomez-Perez
Pierre Grenon
Martin Hepp
Patrick Lambrix
Diana Maynard
Riichiro Mizoguchi
Viktoria Pammer
Hsofia Pinto
Dimitris Plexousakis
Chantal Reynaud
Marta Sabou
Ansgar Scherp
Guus Schreiber
Luciano Serafini
Michael Sintek
Robert Stevens
Vojtech Svatek
Johanna Voelker

Program Committee - Reasoning Track

Track Chairs

Emanuele Della Valle and Pascal Hitzler

Members

Jose Julio Alferes
Jie Bao
Andre Bolles
Daniele Braga
Diego Calvanese
Irene Celino
Oscar Corcho
Bernardo Cuenca Grau
Claudia D'Amato
Mathieu D'Aquin
Daniele Dell'Aglio
Michael Grossniklaus
Rinke Hoekstra
Zhisheng Huang
Markus Krötzsch
Thomas Lukasiewicz
Marko Luther
Frederick Maier
Jeff Z. Pan
Bijan Parsia
Guilin Qi
Dumitru Roman
Riccardo Rosati
Sebastian Rudolph
Stefan Schlobach
Michael Sintek
Evren Sirin
Annette Ten Teije
Kristin Tufte
Guido Vetere
Zhe Wu
Antoine Zimmermann

Program Committee - Semantic Data Management Track

Track Chairs

Vassilis Christophides and Axel Polleres

Members

Karl Aberer
Abraham Bernstein
Aidan Boran
Alessandro Bozzon
Stefano Ceri
Oscar Corcho
Orri Erling
George H. L. Fletcher
Irini Fundulaki
Claudio Gutierrez
Steve Harris
Andreas Harth
Aidan Hogan
Marcel Karnstedt
Panagiotis Karras
Greg Karvounarakis
Anastasios Kementsietsidis
Atanas Kiryakov
Dimitris Kotzinos
Manolis Koubarakis
Reto Krummenacher
Georg Lausen
Josiane Xavier Parreira
Sherif Sakr
Andy Seaborne
Amit Sheth
Umberto Straccia
Letizia Tanca
Thanh Tran
Giovanni Tummarello
Jacopo Urbani
Yannis Velegrakis
Maria Esther Vidal
Jesse Weaver

Program Committee - Semantic Web in Use Track

Track Chairs

Daniel Olmedilla Pavel Shvaiko

Members

Harith Alani
George Anadiotis
Giuseppe Angelini
SÃren Auer
Stefano Bertolo
Olivier Bodenreider
Paolo Bouquet
François Bry
Pablo Castells
John Davies
Mike Dean
Lee Feigenbaum
Aldo Gangemi
Fausto Giunchiglia
John Goodwin
Peter Haase
Bin He
Tom Heath
Nicola Henze
Ivan Herman
Geert-Jan Houben
Eero Hyvönen
Renato Iannella
Antoine Isaac
Alexander Ivanyukovich
Krzysztof Janowicz

Yannis Kalfoglou
Atanas Kiryakov
Birgitta König-Ries
Rubén Lara
Nico Lavarini
Alain Leger
Maurizio Lenzerini
Bernardo Magnini
Vincenzo Maltese
Massimo Marchiori
Peter Mika
Luca Mion
Andriy Nikolov
Lyndon Nixon
Leo Obrst
Massimo Paolucci
Yefei Peng
Erhard Rahm
Yves Raimond
Sebastian Schaffert
Hannes Schwetz
Kavitha Srinivas
Andrei Tamilin
Klaus-Dieter Thoben
Andraz Tori
Tania Tudorache
Lorenzino Vaccari
Michael Witbrock
Baoshi Yan
Ilya Zaihrayeu
Songmao Zhang

Program Committee - Sensor Web Track

Track Chairs

Harith Alani and Luca Mottola

Members

Philippe Bonnet
Ciro Cattuto
Oscar Corcho
David De Roure
Cory Henson
Krzysztof Janowicz
Yong Liu
Pedro Jose Marron
Kirk Martinez
Josiane Xavier Parreira
Eric Pauwels
Jamie Payton
Vinny Reynolds
Mark Roantree
Kay Roemer
Nenad Stojanovic
Kerry Taylor
Eiko Yoneki

Program Committee - Software, Services, Processes and Cloud Computing Track

Track Chairs

Barry Norton and Michael Stollberg

Members

Sudhir Agarwal
Luciano Baresi
Irene Celino
Violeta Damjanovic
Pieter De Leenheer
Tommaso Di Noia

Federico Facca
José Fiadeiro
Agata Filipowska
Dragan Gasevic
Stephan Grimm
Dimka Karastoyanova
Holger Kett
Reto Krummenacher
Dave Lambert
Florian Lautenbacher
Niels Lohmann
Jan Mendling
Andreas Metzger
Massimo Paolucci
Carlos Pedrinaci
Pierluigi Plebani
Dumitru Roman
Monika Solanki
Nenad Stojanovic
Ioan Toma
Jürgen Vogel

Program Committee - Social Web and Web Science Track

Track Chairs

Alexandre Passant and Denny Vrandecic

Members

Fabian Abel
Harith Alani
Sören Auer
Scott Bateman
Edward Benson
Shlomo Berkovsky
John Breslin
Ciro Cattuto
Federica Cena
Richard Cyganiak
Antonina Dattolo
Darina Dicheva
Ying Ding
Jon Dron
Guillaume Ereteo
Anna Fensel
Fabien Gandon
Cristina Gena
Steve Harris
Aidan Hogan
Ekaterini Ioannou
Neil Ireson
Robert Jaschke
Lalana Kagal
Pranam Kolari
Georgia Koutrika
Milos Kravcik
Juanzi Li
Meenakshi Nagarajan
Matthew Rowe
Ansgar Scherp
Juan F. Sequeda
Paul Smart
Sergey Sosnovsky
Steffen Staab
Markus Strohmaier
Christopher Thomas
Mischa Tuffield
Shenghui Wang
Katrin Weller
Mary-Anne Williams
Jie Zhang
Lina Zhou

Program Committee - PhD Symposium

Track Chairs

Pieter Deleenheer and Jeff Z. Pan

Members

Diego Calvanese
Bernardo Cuenca Grau
Mathieu D'Aquin
Jianfeng Du
Giorgos Flouris
Tim Furche
Zhiqiang Gao
Pascal Hitzler
Laura Hollink
Zhisheng Huang
Juanzi Li
Diana Maynard
Jing Mei
Ralf Möller
Dimitris Plexousakis
Guilin Qi
Alan Ruttenberg
Manuel Salvadores
Kai-Uwe Sattler
Stefan Schlobach
Murat Sensoy
Luciano Serafini
Yi-Dong Shen
Kavitha Srinivas
Giorgos Stoilos
Heiner Stuckenschmidt
Vassilis Tzouvaras
Haofen Wang
Kewen Wang
Shenghui Wang
Benjamin Zapilko
Ming Zhang
Yuting Zhao

Referees

Sofia Angeletou
Darko Anicic
Fedor Bakalov
Jürgen Bock
Stefano Bortoli
Sebastian Brandt
Siarhei Bykau
Michele Caci
Elena Cardillo
Michele Catasta
Gong Cheng
Annamaria Chiasera
Catherine Comparot
Gianluca Correndo
Brian Davis
Evangelia Daskalaki
Renaud Delbru
Kathrin Dentler
Huyen Do
Vicky Dritsou
George Eadon
Angela Fogarolli
Anika Gross
Karl Hammar
Matthias Hert
Martin Homola
Matthew Horridge
Wei Hu
Prateek Jain
Mouna Kamel
Malte Kiesel
Szymon Klarman
Johannes Knopp
Matthias Knorr
Haridimos Kondylakis
Jacek Kopecky
Günter Ladwig
Feiyu Lin
Dong Liu
Christian Meilicke
Ivan Mikhailov
Rammohan Narendula
Axel-Cyrille Ngonga Ngomo
Maximilian Nickel
Andriy Nikolov
Dusan Omercevic
Giorgio Orsi
Matteo Palmonari
Bastien Rance
Mariano Rodriguez Muro
Marco Rospocher
Brigitte Safar

Fatiha Sais	Dimitrios Skoutas	Alejandro Vaisman
Frank Sawitzki	Philipp Sorg	Andreas Wagner
Philipp Schaer	Sebastian Speiser	Claudia Wagner
Anne Schlicht	Florian Steinke	Fang Wei
Eric Schmieders	Cosmin Stroe	Pia-Ramona Wojtinnek
Michael Schmidt	Ondrej Svab-Zamazal	Gang Wu
Luigi Selmi	Stuart Taylor	Honghan Wu
Gerardo Simari	Edward Thomas	Surender Reddy Yerva
Sergej Sizov	Kateryna Thymoshenko	Benjamin Zapilko

Steering Committee

Chair

John Domingue

Members

Lora Aroyo	Eero Hyvönen
Sean Bechhofer	Michael Kifer
Fabio Ciravegna	Paolo Traverso
Enrico Franconi	

Sponsoring Institutions

Platinum Sponsors

ontotext

Gold Sponsors

COIN Enterprise COllaboration & INteroperability

ELSEVIER

insemtives
Incentives for Semantics

Planet Data

Silver Sponsors

Best Paper Award Sponsors

Video Recording Sponsors

Table of Contents – Part I

Digital Libraries Track

Interactive Exploration of Fuzzy RDF Knowledge Bases 1
Nikos Manolis and Yannis Tzitzikas

A Structured Semantic Query Interface for Reasoning-Based Search and Retrieval 17
Dimitrios A. Koutsomitropoulos, Ricardo Borillo Domenech, and Georgia D. Solomou

Distributed Human Computation Framework for Linked Data Co-reference Resolution 32
Yang Yang, Priyanka Singh, Jiadi Yao, Ching-man Au Yeung, Amir Zareian, Xiaowei Wang, Zhonglun Cai, Manuel Salvadores, Nicholas Gibbins, Wendy Hall, and Nigel Shadbolt

Inductive and Probabilistic Approaches Track

Relational Kernel Machines for Learning from Graph-Structured RDF Data 47
Veli Bicer, Thanh Tran, and Anna Gossen

AutoSPARQL: Let Users Query Your Knowledge Base 63
Jens Lehmann and Lorenz Bühmann

Contextual Ontology Alignment of LOD with an Upper Ontology: A Case Study with Proton 80
Prateek Jain, Peter Z. Yeh, Kunal Verma, Reymonrod G. Vasquez, Mariana Damova, Pascal Hitzler, and Amit P. Sheth

Linked Open Data Track

Hide the Stack: Toward Usable Linked Data 93
Aba-Sah Dadzie, Matthew Rowe, and Daniela Petrelli

Linked Data Metrics for Flexible Expert Search on the open Web 108
Milan Stankovic, Jelena Jovanovic, and Philippe Laublet

Statistical Schema Induction 124
Johanna Völker and Mathias Niepert

SIHJoin: Querying Remote and Local Linked Data 139
Günter Ladwig and Thanh Tran

Zero-Knowledge Query Planning for an Iterator Implementation of Link Traversal Based Query Execution 154
Olaf Hartig

Integrating Linked Data and Services with Linked Data Services 170
Sebastian Speiser and Andreas Harth

Mobile Web Track

OntoWiki Mobile – Knowledge Management in Your Pocket 185
Timofey Ermilov, Norman Heino, Sebastian Tramp, and Sören Auer

Weaving a Distributed, Semantic Social Network for Mobile Users 200
Sebastian Tramp, Philipp Frischmuth, Natanael Arndt, Timofey Ermilov, and Sören Auer

Natural Language Processing Track

Automatic Semantic Subject Indexing of Web Documents in Highly Inflected Languages 215
Reetta Sinkkilä, Osma Suominen, and Eero Hyvönen

FootbOWL: Using a Generic Ontology of Football Competition for Planning Match Summaries 230
Nadjet Bouayad-Agha, Gerard Casamayor, Leo Wanner, Fernando Díez, and Sergio López Hernández

Linking Lexical Resources and Ontologies on the Semantic Web with Lemon 245
John McCrae, Dennis Spohr, and Philipp Cimiano

Ontologies Track

Elimination of Redundancy in Ontologies 260
Stephan Grimm and Jens Wissmann

Evaluating the Stability and Credibility of Ontology Matching Methods 275
Xing Niu, Haofen Wang, Gang Wu, Guilin Qi, and Yong Yu

How Matchable Are Four Thousand Ontologies on the Semantic Web ... 290
Wei Hu, Jianfeng Chen, Hang Zhang, and Yuzhong Qu

Understanding an Ontology through Divergent Exploration............ 305
Kouji Kozaki, Takeru Hirota, and Riichiro Mizoguchi

The Use of Foundational Ontologies in Ontology Development: An Empirical Assessment 321
C. Maria Keet

Using Pseudo Feedback to Improve Cross-Lingual Ontology Mapping ... 336
Bo Fu, Rob Brennan, and Declan O'Sullivan

Automatic Identification of Ontology Versions Using Machine Learning Techniques 352
Carlo Allocca

Reasoning Track

A Tableaux-Based Algorithm for $\mathcal{SHIQ}$ with Transitive Closure of Roles in Concept and Role Inclusion Axioms 367
Chan Le Duc, Myriam Lamolle, and Olivier Curé

SPARQL Query Answering over OWL Ontologies 382
Ilianna Kollia, Birte Glimm, and Ian Horrocks

Epistemic Querying of OWL Knowledge Bases 397
Anees Mehdi, Sebastian Rudolph, and Stephan Grimm

A Practical Approach for Computing Generalization Inferences in $\mathcal{EL}$ 410
Rafael Peñaloza and Anni-Yasmin Turhan

Author Index 425

Table of Contents – Part II

Semantic Data Management Track

Semantics and Optimization of the SPARQL 1.1 Federation Extension 1
Carlos Buil-Aranda, Marcelo Arenas, and Oscar Corcho

Grr: Generating Random RDF 16
Daniel Blum and Sara Cohen

High-Performance Computing Applied to Semantic Databases 31
Eric L. Goodman, Edward Jimenez, David Mizell, Sinan al-Saffar, Bob Adolf, and David Haglin

An Intermediate Algebra for Optimizing RDF Graph Pattern Matching on MapReduce 46
Padmashree Ravindra, HyeongSik Kim, and Kemafor Anyanwu

Query Relaxation for Entity-Relationship Search 62
Shady Elbassuoni, Maya Ramanath, and Gerhard Weikum

Optimizing Query Shortcuts in RDF Databases 77
Vicky Dritsou, Panos Constantopoulos, Antonios Deligiannakis, and Yannis Kotidis

RDFS Update: From Theory to Practice 93
Claudio Gutierrez, Carlos Hurtado, and Alejandro Vaisman

Benchmarking Matching Applications on the Semantic Web 108
Alfio Ferrara, Stefano Montanelli, Jan Noessner, and Heiner Stuckenschmidt

Efficiently Evaluating Skyline Queries on RDF Databases 123
Ling Chen, Sidan Gao, and Kemafor Anyanwu

The Design and Implementation of Minimal RDFS Backward Reasoning in 4store 139
Manuel Salvadores, Gianluca Correndo, Steve Harris, Nick Gibbins, and Nigel Shadbolt

Semantic Web in Use Track

miKrow: Semantic Intra-enterprise Micro-Knowledge Management System 154
Víctor Penela, Guillermo Álvaro, Carlos Ruiz, Carmen Córdoba, Francesco Carbone, Michelangelo Castagnone, José Manuel Gómez-Pérez, and Jesús Contreras

A Faceted Ontology for a Semantic Geo-catalogue 169
Feroz Farazi, Vincenzo Maltese, Fausto Giunchiglia, and Alexander Ivanyukovich

SoKNOS – Using Semantic Technologies in Disaster Management Software 183
Grigori Babitski, Simon Bergweiler, Olaf Grebner, Daniel Oberle, Heiko Paulheim, and Florian Probst

Semantic Technologies for Describing Measurement Data in Databases 198
Ulf Noyer, Dirk Beckmann, and Frank Köster

Ontology-Driven Guidance for Requirements Elicitation 212
Stefan Farfeleder, Thomas Moser, Andreas Krall, Tor Stålhane, Inah Omoronyia, and Herbert Zojer

The Semantic Public Service Portal (S-PSP) 227
Nikolaos Loutas, Deirdre Lee, Fadi Maali, Vassilios Peristeras, and Konstantinos Tarabanis

DataFinland—A Semantic Portal for Open and Linked Datasets 243
Matias Frosterus, Eero Hyvönen, and Joonas Laitio

Biological Names and Taxonomies on the Semantic Web – Managing the Change in Scientific Conception 255
Jouni Tuominen, Nina Laurenne, and Eero Hyvönen

An Approach for More Efficient Energy Consumption Based on Real-Time Situational Awareness 270
Yongchun Xu, Nenad Stojanovic, Ljiljana Stojanovic, Darko Anicic, and Rudi Studer

Sensor Web Track

Ontology-Driven Complex Event Processing in Heterogeneous Sensor Networks 285
Kerry Taylor and Lucas Leidinger

A Semantically Enabled Service Architecture for Mashups over Streaming and Stored Data 300
Alasdair J.G. Gray, Raúl García-Castro, Kostis Kyzirakos, Manos Karpathiotakis, Jean-Paul Calbimonte, Kevin Page, Jason Sadler, Alex Frazer, Ixent Galpin, Alvaro A.A. Fernandes, Norman W. Paton, Oscar Corcho, Manolis Koubarakis, David De Roure, Kirk Martinez, and Asunción Gómez-Pérez

Software, Services, Processes and Cloud Computing Track

Zhi# – OWL Aware Compilation 315
Alexander Paar and Denny Vrandečić

Lightweight Semantic Annotation of Geospatial RESTful Services 330
Víctor Saquicela, Luis.M. Vilches-Blazquez, and Oscar Corcho

Towards Custom Cloud Services: Using Semantic Technology to Optimize Resource Configuration 345
Steffen Haak and Stephan Grimm

Social Web and Web Science Track

One Tag to Bind Them All: Measuring Term Abstractness in Social Metadata 360
Dominik Benz, Christian Körner, Andreas Hotho, Gerd Stumme, and Markus Strohmaier

Semantic Enrichment of Twitter Posts for User Profile Construction on the Social Web 375
Fabian Abel, Qi Gao, Geert-Jan Houben, and Ke Tao

Improving Categorisation in Social Media Using Hyperlinks to Structured Data Sources 390
Sheila Kinsella, Mengjiao Wang, John G. Breslin, and Conor Hayes

Predicting Discussions on the Social Semantic Web 405
Matthew Rowe, Sofia Angeletou, and Harith Alani

Mining for Reengineering: An Application to Semantic Wikis Using Formal and Relational Concept Analysis 421
Lian Shi, Yannick Toussaint, Amedeo Napoli, and Alexandre Blansché

Demo Track

SmartLink: A Web-Based Editor and Search Environment for Linked Services 436
Stefan Dietze, Hong Qing Yu, Carlos Pedrinaci, Dong Liu, and John Domingue

ViziQuer: A Tool to Explore and Query SPARQL Endpoints 441
Martins Zviedris and Guntis Barzdins

EasyApp: Goal-Driven Service Flow Generator with Semantic Web Service Technologies 446
Yoo-mi Park, Yuchul Jung, HyunKyung Yoo, Hyunjoo Bae, and Hwa-Sung Kim

Who's Who – A Linked Data Visualisation Tool for Mobile Environments 451
A. Elizabeth Cano, Aba-Sah Dadzie, and Melanie Hartmann

OntosFeeder – A Versatile Semantic Context Provider for Web Content Authoring 456
Alex Klebeck, Sebastian Hellmann, Christian Ehrlich, and Sören Auer

wayOU – Linked Data-Based Social Location Tracking in a Large, Distributed Organisation 461
Mathieu d'Aquin, Fouad Zablith, and Enrico Motta

SeaFish: A Game for Collaborative and Visual Image Annotation and Interlinking 466
Stefan Thaler, Katharina Siorpaes, David Mear, Elena Simperl, and Carl Goodman

The Planetary System: Executable Science, Technology, Engineering and Math Papers 471
Christoph Lange, Michael Kohlhase, Catalin David, Deyan Ginev, Andrea Kohlhase, Bogdan Matican, Stefan Mirea, and Vyacheslav Zholudev

Semantic Annotation of Images on Flickr 476
Pierre Andrews, Sergey Kanshin, Juan Pane, and Ilya Zaihrayeu

FedX: A Federation Layer for Distributed Query Processing on Linked Open Data 481
Andreas Schwarte, Peter Haase, Katja Hose, Ralf Schenkel, and Michael Schmidt

PhD Symposium

Reasoning in Expressive Extensions of the RDF Semantics 487
Michael Schneider

Personal Semantics: Personal Information Management in the Web with Semantic Technologies 492
Salman Elahi

Reasoning with Noisy Semantic Data 497
Qiu Ji, Zhiqiang Gao, and Zhisheng Huang

Extracting and Modeling Historical Events to Enhance Searching and Browsing of Digital Cultural Heritage Collections 503
Roxane Segers

Enriching Ontologies by Learned Negation: Or How to Teach Ontologies Vegetarianism 508
Daniel Fleischhacker

Optimizing Query Answering over OWL Ontologies 513
Ilianna Kollia

Hybrid Search Ranking for Structured and Unstructured Data 518
Daniel M. Herzig

Author Index 523

Interactive Exploration of Fuzzy RDF Knowledge Bases

Nikos Manolis and Yannis Tzitzikas

Institute of Computer Science, FORTH-ICS, Greece, and
Computer Science Department, University of Crete, Greece
{manolisn,tzitzik}@ics.forth.gr

Abstract. In several domains we have objects whose descriptions are accompanied by a degree expressing their strength. Such degrees can have various application specific semantics, such as relevance, precision, certainty, trust, etc. In this paper we consider *Fuzzy RDF* as the representation framework for such "weighted" descriptions and associations, and we propose a novel model for *browsing and exploring* such sources, which allows formulating complex queries gradually and through plain clicks. Specifically, and in order to exploit the fuzzy degrees, the model proposes interval-based transition markers. The advantage of the model is that it significantly increases the discrimination power of the interaction, without making it complex for the end user.

1 Introduction

There are many practical situations where the descriptions of objects are accompanied by a degree. These degrees can be provided by humans or be the result of automated tasks. For instance, in [19] they express the degree of a feature that is extracted from data, in [7] they express the certainty of membership of a document to an aspect (automatically retrieved from the answer set of a keyword query), in [17] the membership of an object to a cluster, in [8] the evaluation scores of products under hierarchically organized criteria, in [13] they express the frequency of symptoms in a disease. Furthermore in an open environment like the Web, we may have data of various degrees of credibility, as well data which are copies or modifications of other data. As a result, data of the same entity can be erroneous, out-of-date, or inconsistent/conflicting in different data sources. Therefore, even if the primary data are not fuzzy, the integrated data as produced by an information integration system (that contains tasks like web extraction) can be fuzzy. Synopsizing, such degrees can capture various application specific semantics, such as *relevance*, *precision*, *certainty*, *trust*, etc.

Users would like to browse and explore such sources without having to be aware of the terminology, contents or query language of the source. Furthermore, the fuzzy degrees should be exploitable, allowing the users to reach states which are characterized by conditions that involve degrees. Finally, it should be possible to offer adequate support for recall-oriented information needs, and support a

© Springer-Verlag Berlin Heidelberg 2011

gradual focus restriction process (i.e. interactive search). However there is not any exploration/browsing approach for such sources.

The contribution of this paper lies in introducing a model for exploring such sources. The merit of the proposed model is that it defines formally and precisely the state space of an interaction that (a) allows users to locate the objects of interest, or to get overviews, without having to be aware of the terminology nor the query language of the underlying source, and without reaching states with empty results, (b) exploits fuzzy degrees for enhancing the discrimination power of the interaction, (c) generalizes the main exploration/browsing approaches for plain RDF/S sources (also clarifying issues regarding schema and instance cyclic property paths), (d) is query language independent, and (e) is visualization independent. Finally it discusses issues that concern the available query languages.

To grasp the idea, Fig. 1 shows an example of a Fuzzy RDF KB where instance triples have fuzzy degrees. Properties are depicted by rectangles and the letters "d" and "r" denote the domain and the range of a property. Fat arrows denote subClassOf/subPropertyOf relationships, while dashed arrows denote instanceOf relationships. Note that various approaches have been proposed recently for annotating triples with a degree of truth and giving a semantics to such annotations, e.g. [16,18], and can be used as the representation framework. Among the various possible approaches for exploiting fuzzy degrees at the interaction level, we propose one based on *intervals*, leading to a simple and intuitive dialog. The idea is to analyze the count information of each transition marker to more than one counts each corresponding to the objects whose membership degrees fall into a specified interval. An instance of the proposed interaction is sketched at Fig. 2. The figure depicts only the part of the UI (usually the left bar) that shows the transition markers (it does not show the object set). Fig. 2(a) ignores fuzzy degrees. Fig. 2(b) shows how fuzzy degrees are exploited. For example, consider a transition marker "z(20)". We can present it as "z [low(10), medium(4), high(6)]" where "low" may correspond to degrees (0,0.3], "medium" to (0.3,0.6] and "high" to (0.6,1]. So the user has now three clicks (i.e. three transitions) instead of one. Therefore the set of all possible foci is more, so the discrimination power of interaction increases. The increased discrimination power is useful especially in cases where the fuzzy degrees are derived by automatic methods. Since the results of such methods are vulnerable to errors and inaccuracies (i.e. the derived degrees may be lower or higher than those deserved), it would not be wise to exclude the descriptions having low degrees. Instead, it is better to make all of them available and let the user free to explore also the objects having low degrees if necessary. In brief the interaction leads to states whose extension corresponds to the answer of complex path expressions that involve interval-based conditions over the fuzzy degrees, e.g. queries of the form: *" $\underline{\text{Japanese cars}}$ $\underline{\text{for sale}}$ which are $\underline{\text{driven by}}$ persons $\underline{\text{who work at FORTH}}$ and $\underline{\text{know a person}}$ who $\underline{\text{knows Bob}}$",* where each individual underlined part (value-based condition) is associated also with an interval-based condition over the fuzzy degrees.

The paper is organized as follows. Section 2 describes in brief related works. Section 3 introduces the least number of symbols and notations required for

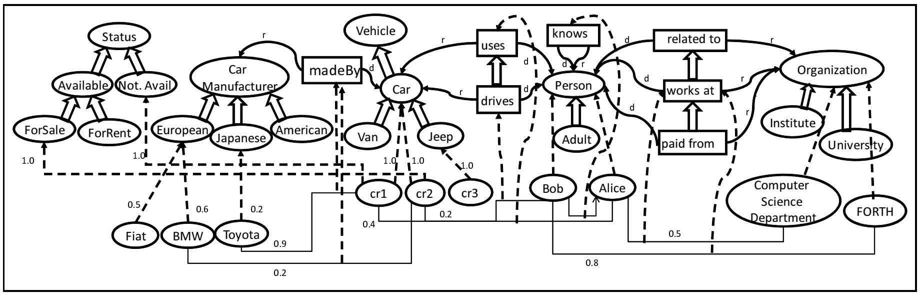
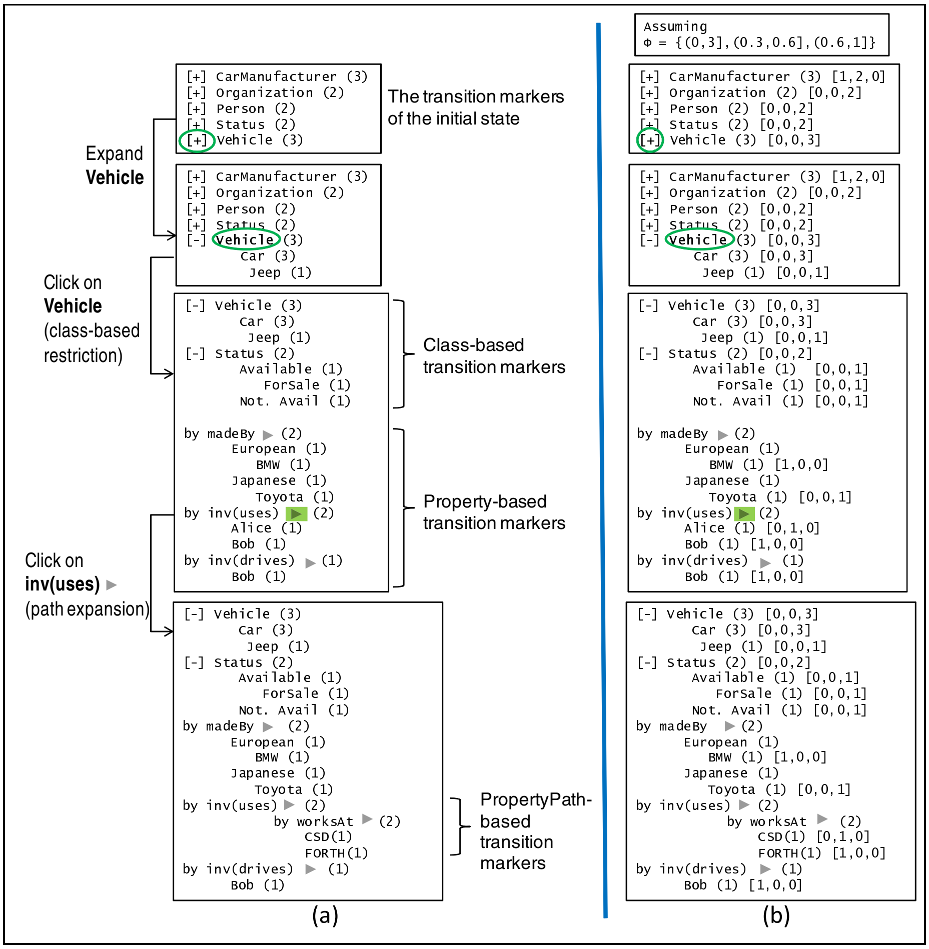


Fig. 1. A Fuzzy RDF KB


Expand **Vehicle**
The transition markers of the initial state
Click on **Vehicle** (class-based restriction)
Class-based transition markers
Property-based transition markers
Click on **inv(uses)** ▶ (path expansion)
PropertyPath-based transition markers
(a)
(b)

```
[+] CarManufacturer (3)
[+] Organization (2)
[+] Person (2)
[+] Status (2)
[+] Vehicle (3)
```

```
[+] CarManufacturer (3)
[+] Organization (2)
[+] Person (2)
[+] Status (2)
[-] Vehicle (3)
      Car (3)
       Jeep (1)
```

```
[-] Vehicle (3)
      Car (3)
       Jeep (1)
[-] Status (2)
      Available (1)
         ForSale (1)
      Not. Avail (1)

by madeBy ▶ (2)
     European (1)
        BMW (1)
     Japanese (1)
        Toyota (1)
by inv(uses) ▶ (2)
     Alice (1)
     Bob (1)
by inv(drives) ▶ (1)
     Bob (1)
```

```
[-] Vehicle (3)
      Car (3)
       Jeep (1)
[-] Status (2)
      Available (1)
         ForSale (1)
      Not. Avail (1)
by madeBy ▶ (2)
     European (1)
        BMW (1)
     Japanese (1)
        Toyota (1)
by inv(uses) ▶ (2)
              by worksAt ▶ (2)
                 CSD(1)
                 FORTH(1)
by inv(drives) ▶ (1)
     Bob (1)
```

```
Assuming
Φ = {(0,3],(0.3,0.6],(0.6,1]}
```

```
[+] CarManufacturer (3) [1,2,0]
[+] Organization (2) [0,0,2]
[+] Person (2) [0,0,2]
[+] Status (2) [0,0,2]
[+] Vehicle (3) [0,0,3]
```

```
[+] CarManufacturer (3) [1,2,0]
[+] Organization (2) [0,0,2]
[+] Person (2) [0,0,2]
[+] Status (2) [0,0,2]
[-] Vehicle (3) [0,0,3]
      Car (3) [0,0,3]
       Jeep (1) [0,0,1]
```

```
[-] Vehicle (3) [0,0,3]
      Car (3) [0,0,3]
       Jeep (1) [0,0,1]
[-] Status (2) [0,0,2]
      Available (1)  [0,0,1]
         ForSale (1) [0,0,1]
      Not. Avail (1) [0,0,1]

by madeBy ▶ (2)
     European (1)
        BMW (1) [1,0,0]
     Japanese (1)
        Toyota (1) [0,0,1]
by inv(uses) ▶ (2)
     Alice (1) [0,1,0]
     Bob (1) [1,0,0]
by inv(drives) ▶ (1)
     Bob (1) [1,0,0]
```

```
[-] Vehicle (3) [0,0,3]
      Car (3) [0,0,3]
       Jeep (1) [0,0,1]
[-] Status (2) [0,0,2]
      Available (1) [0,0,1]
         ForSale (1) [0,0,1]
      Not. Avail (1) [0,0,1]
by madeBy ▶ (2)
     European (1)
        BMW (1) [1,0,0]
     Japanese (1)
        Toyota (1) [0,0,1]
by inv(uses) ▶ (2)
              by worksAt ▶ (2)
                 CSD(1) [0,1,0]
                 FORTH(1) [1,0,0]
by inv(drives) ▶ (1)
     Bob (1) [1,0,0]
```

Fig. 2. Sketch of the GUI part for transition markers
(a): ignores fuzzy degrees. (b): count information based on intervals over the fuzzy degrees

defining the interaction model. Section 4 introduces the model first for plain RDF/S and then extends it for Fuzzy RDF. Section 5 discusses query language issues, and finally Section 6 concludes the paper.

2 Related Work

Some browsing approaches are applicable to simple structures (like attribute-value pairs), while others to complex information structures (e.g. OWL-based KBs). Therefore one important aspect is how the underlying information is structured. There are several options, some of them follow: attribute-value pairs with flat values (e.g. `name=Yannis`), attribute-value pairs with hierarchically organized values (e.g. `location=Crete`), set-valued attributes (either flat or hierarchical) (e.g. `accessories={ABS, ESP}`), multi-entity (or object-oriented) (e.g. RDF, Linked Open Data), and relational databases. Furthermore, we could have fuzziness and we can consider this as an independent aspect (e.g. there are fuzzy extensions of the RDF model such as [10,16]).

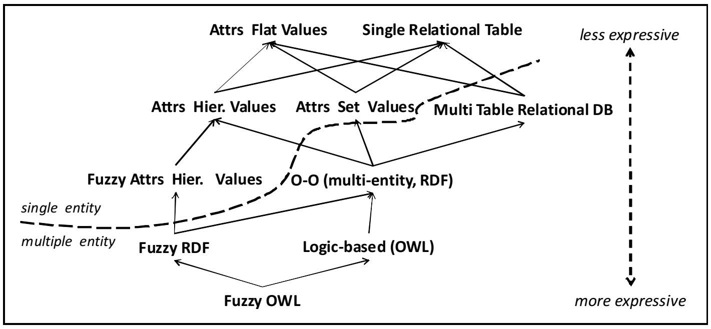


Fig. 3. Categories of Information Spaces

Fig. 3 shows the above categories organized hierarchically where an option X is a (direct or indirect) child of an option Y if whatever information can be expressed in Y can also be expressed in X. The value of this diagram is that it depicts the fact that if a browsing approach is appropriate for an option X, then certainly it is appropriate for all options which are parents of X. For instance, a browsing approach appropriate for Fuzzy RDF is also appropriate for plain RDF, as well as for sources formed by attributes with fuzzy and hierarchically organized values. Just indicatively, the following table lists some browsing approaches grouped according to their applicability. In this paper we focus on Fuzzy RDF and we are interested in generic approaches (not requiring special configuration), that exploit schema information (if available). In contrast, [6] deals with fuzzy annotations of documents w.r.t. a particular ontology using a view-based approach where the browsable elements (views) are predefined by specialists.

Information Space	System
Attrs with flat values	Elastic Lists [14]
Attrs with hierarchical values	Flamenco [20], Mitos WSE [12]
Object-Oriented (e.g. RDF)	GRQL [2], BrowseRdf[11], Ontogator [9], VisiNav [4]
FRDF (Fuzzy RDF)	Fuzzy view-based Semantic Search [6]
OWL	Odalisque [1]

3 Fuzzy RDF

Consider a set of RDF triples K and let $\mathcal{C}(K)$ be its closure. We shall denote with C the set of classes, with Pr the set of properties, with $\leq_{cl}$ the subclassOf relation between classes, and with $\leq_{pr}$ the subpropertyOf relation between properties. The instances of a class $c \in C$ are $inst(c) = \{o \mid (o, type, c) \in \mathcal{C}(K)\}$, while the instances of a property $p \in Pr$ are $inst(p) = \{\ (o, p, o') \mid (o, p, o') \in \mathcal{C}(K)\}$. Now we introduce some notations for Fuzzy RDF. Each instance triple t (either class or property instance triple) is accompanied by a fuzzy degree, which we shall denote by $directdegree(t)$. We can now define the degree of t, denoted by $degree(t)$, based on the semantics of RDF, and the axioms of Fuzzy Set Theory[1]. Specifically,
$degree(o, type, c) = \max\{\ directdegree(o, type, c') \mid c' \leq_{cl} c\}$
$degree(o, p, o') = \max\{\ directdegree(o, p', o') \mid p' \leq_{pr} p\}$.
Let $\Phi = \{\varphi_1, \ldots \varphi_m\}$ be a set of intervals in [0,1]. We define:
$inst(c, \varphi) = \{\ o \in inst(c) \mid degree(o, type, c) \in \varphi\}$
$inst(p, \varphi) = \{\ (o, p, o') \in inst(p) \mid degree(o, p, o') \in \varphi\}$.

4 The Interaction Model

The interaction is modeled by a *state space*. Each *state* has an *extension* and a number of *transitions* leading to other states. Each transition is signified by a *transition marker* accompanied by a number showing the size of the extension of the targeting state (we will refer to this with *count* information). Such view abstracts from the various visualization approaches; in general each state has one or more visualization modes for its extension as well as its transition markers.

4.1 The Interaction Model for Plain RDF/S

The objective is to define a precise and concise model capturing the essentials of RDF browsing approaches, which later will be extended to capture Fuzzy RDF.

Consider that we are in the context of one RDF/S KB with a single namespace with classes C and properties Pr. If s denotes a state we shall use $s.e$ to denote its extension. Let's start from the *initial state(s)*. Let s_0 denote an artificial initial state. We can assume that $s_0.e = URI \cup LIT$, i.e. its extension contains every URI and literal of the KB. Alternatively, the extension of the initial state can

[1] We will adopt Zadeh's theory and consequently we shall use min/max, however one could also adopt alternative definitions for the operators $\bigotimes$, $\bigoplus$ (e.g. as done in [16]).

be the result of a keyword query, or a set of resources provided by an external access method. Given a state we shall show how to compute the transitions that are available to that state. From s_0 the user can move to states corresponding to the maximal classes and properties, i.e. to one state for each $maximal_{\leq_{cl}}(C)$ and each $maximal_{\leq_{pr}}(Pr)$. Specifically, each $c \in maximal_{\leq_{cl}}(C)$ (resp. $p \in maximal_{\leq_{pr}}(Pr)$) yields a state with extension $inst(c)$ (resp. $inst(p)$).

We will define formally the transitions based on the notion of *restriction* and *join*. To this end we introduce some auxiliary definitions. We shall use p^{-1} to denote the inverse direction of a property p, e.g. if $(d, p, r) \in Pr$ then $p^{-1} = (r, inv(p), d)$, and let Pr^{-1} denote the inverse properties of all properties in Pr. If E is a set of resources, p is a property in Pr or Pr^{-1}, v is a resource or literal, $vset$ is a set of resources or literals, and c is a class, we define the following notations for *restricting* the set E:

$$\begin{aligned} Restrict(E, p : v) &= \{\ e \in E \mid (e, p, v) \in inst(p)\} \\ Restrict(E, p : vset) &= \{\ e \in E \mid \exists\ v' \in vset \text{ and } \ (e, p, v') \in inst(p)\} \\ Restrict(E, c) &= \{\ e \in E \mid e \in inst(c)\} \end{aligned}$$

Now we define a notation for *joining* values, i.e. for computing values which are linked with the elements of E:

$$Joins(E, p) = \{\ v \mid \exists\ e \in E \text{ and } \ (e, p, v) \in inst(p)\}$$

We can now define precisely transitions and transition markers. Suppose we are in a state s with extension $s.e$.

Class-based browsing. The classes that can be used as class-based transition markers, denoted by $TM_{cl}(E)$, are defined by:

$$TM_{cl}(E) = \{c \in C \mid Restrict(E, c) \neq \emptyset\} \tag{1}$$

If the user clicks on a $c \in TM_{cl}(s.e)$, then the extension of the targeting state s' is defined as $s'.e = Restrict(s.e, c)$, and its count information is $s'.count = |s'.e|$. For example, suppose the user selects the class `Vehicle`. The user can then view its instances and follow one of the following class-based transition markers: `Vehicle, Car, Jeep, Status, Available, ForSale, Not.Available`. Notice that `ForRent` and `Van` are not included because their extension (and thus their intersection with the current extension) is empty.

The elements of $TM_{cl}(s.e)$ can be hierarchically organized (based on the subclass relationships among them). Specifically the layout (e.g. the indentation in a text-based layout) of the transition markers can be based on the relationships of the reflexive and transitive reduction of the restriction of $\leq_{cl}$ on $TM_{cl}(s.e)$ (i.e. on $R^{refl,trans}(\leq_{cl} \mid_{TM_{cl}(s.e)})$). In our case, we can get what is shown in Fig. 4(a). Furthermore based on the relationship between the extensions $s.e$ and $s'.e$ a transition (or transition marker) can be characterized as a zoom-in/out/side transition.

Let's now focus on ***property-based browsing***. Suppose the user has focused on `Car`, and the extension of this state is $\{$`cr1, cr2, cr3`$\}$. He can further restrict the extension also through the properties. Roughly each property whose domain or range is the class `Car`, or a superclass of `Car` (in general any property that used in the resources in $s.e$), can be considered as a facet of the instances of `Car`. For example, consider a property `madeBy` whose domain is the class `Car` and suppose its range was the `String Literal` class. In that case the firm names of the current extension can be used as transition markers. Now suppose that the range of the property `madeBy` is not literal, but the class `CarManufacturer` (as shown in figure). In this case, the firms (URIs in this case) of the current extension can again be used as transition markers, as shown in Fig. 4(b). Notice that `Fiat` is not shown as it is not related to the current focus (i.e. to `cr1`, `cr2` and `cr3`)[2]. Formally, the properties (in their defined or inverse direction) that can be used for deriving transition markers are defined by:

$$Props(s) = \{p \in Pr \cup Pr^{-1} \mid Joins(s.e, p) \neq \emptyset\} \tag{2}$$

For each $p \in Props(s)$, the corresponding transition markers are defined by $Joins(s.e, p)$, and if the user clicks on a value v in $Joins(s.e, p)$, then the extension of the new state is $s'.e = Restrict(s.e, p : v)$.

```
      (a)              (b)              (c)              (d)                (e)
Vehicle(3)      by madeBy(2)  by madeBy(2)  by inv(uses)(2)   by inv(uses)(2)
  Car(3)           BMW(1)       European(1)     Alice(1)           by worksAt(2)
     Jeep(1)       Toyota(1)      BMW(1)        Bob(1)                CSD(1)
Status(2)                       Japanese(1)  by inv(drives)(1)       FORTH(1)
  Available(1)                    Toyota(1)     Bob(1)
    ForSale(1)
  Not. Avail(1)
```

Fig. 4. Examples of transition markers

Furthermore, the transition markers of a property $p \in Props(s)$, i.e. the set $Joins(s.e, p)$, can be categorized based on their classes. In our example, the firms can be categorized through the subclasses of the class `CarManufacturer`. These classes can be shown as intermediate nodes of the hierarchy that lead to particular car firms, as shown in Fig. 4(c). These classes can be computed easily, they are actually given by $TM_{cl}(Joins(s.e, p))$. Furthermore, these values can be used as *complex transition markers*, i.e. as shortcuts allowing the user to select a set of values with disjunctive interpretation (e.g. he clicks on `Japanese` instead of clicking to every Japanese firm). Specifically, suppose the user clicks on such a value vc. The extension of the target state s' will be:

$$s'.e = Restrict(s.e, p : Restrict(Joins(s.e, p), vc)) \tag{3}$$

[2] Since `cr3` does not participate to a `madeBy` property, an alternative approach is to add an artificial value, like `NonApplicable/Uknown`, whose count would be equal to 1, for informing the user that one element of his focus has not value wrt `madeBy`.

Returning to our example, and while the user has focused on cars, apart from `madeBy`, the user can follow transitions based on the properties `inv(drives)` and `inv(uses)`, as shown in Fig. 4(d). In addition, the elements of $Props(s)$ can be hierarchically organized based on the subProperty relationships among them.

We should be able to extend the above for ***property paths*** of length greater than one. This is needed for restricting the extension through the values of complex attributes (e.g. addresses that may be represented as blank nodes) or through the relationships (direct or indirect) with other resources. For example, one may want to restrict the set of cars so that only cars which are used by persons working for CSD (Computer Science Department) are shown. In that case we would like transition markers of the form shown in Fig. 4(e). It should also be possible the successive "application" of the same property. For example, the user may want to focus to all friends of the friends of `Bob`, or all friends of Bob at distance less than 5. Let's now define precisely, this *property path*-based browsing (expansion and cascading restriction). Let $p_1, \ldots, p_k$ be a sequence of properties. We call this sequence *successive in s* if $Joins(Joins(\ldots(Joins(s.e, p_1), p_2)\ldots p_k) \neq \emptyset$. Obviously such a sequence does not lead to empty results, and can be used to restrict the current focus. Let $M_1, \ldots M_k$ denote the corresponding set of transition markers at each point of the path. Assuming $M_0 = s.e$, the transition markers for all i such that $1 \leq i \leq k$, are defined as:

$$M_i = Joins(M_{i-1}, p_i) \tag{4}$$

What is left to show is how selections on such paths restrict the current focus. Suppose the user selects a value v_k from M_k. This will restrict the set of transitions markers in the following order $M_k, \ldots, M_1$ and finally it will restrict the extension of s. Let $M'_k, \ldots M'_1$ be the restricted set of transitions markers. They are defined as follows: $M'_k = \{v_k\}$, while for each $1 \leq i < k$ we have:

$$M'_i = Restrict(M_i, p_{i+1} : M'_{i+1}) \tag{5}$$

for instance, $M'_1 = Restrict(M_1, p_2 : M'_2)$. Finally, the extension of the new state s' is defined as $s'.e = Restrict(s.e, p_1 : M'_1)$. Equivalently, we can consider that M'_0 corresponds to $s'.e$ and in that case Eq. 5 holds also for $i = 0$.

For example, consider an ontology containing a path of the form:
`Car--hasFirm-->Firm--ofCountry-->Country` and three cars `cr1, cr2, cr3`, the first being `BMW`, the second `VW`, the third `Renault`. The first two firms come from `Germany` the last from `France`. Suppose the user is on `Cars`, and expands the path `hasFirm.ofCountry`. If he selects `Germany`, then the previous list will become `BMW, VW` (so `Renault` will be excluded) and the original focus will be restricted to `cr1` and `cr2`. It follows that path clicks require disjunctive interpretation of the matched values in the intermediate steps.

The above can be applied also for successive applications of the same property, e.g. `inv(drives).knows`^{2}`.paidFrom` is a property path that can be used to restrict cars to those cars whose drivers know some persons who in turn know some persons who are paid from a particular organization.

Entity Type Switch. So far we have described methods to restrict the current extension. Apart from the current extension we can move to other objects. At the simplest case, from one specific resource we move to one resource which is directly or indirectly connected to that. Now suppose that the current focus is a *set* of resources (e.g. cars). Again we can move to one or more resources which are directly or indirectly connected (to all, or at least one) of the resources of the current focus. For example, while viewing a set of cars we can move to (and focus on) the list of their firms (in this way we interpret disjunctively the elements associated with every object of the focus). To capture this requirement it is enough to allow users to move to a state whose extension is the *current set of transition markers.* For example, consider a user who starts from the class `Persons`, and then restricts his focus to those persons who `workAt FORTH`. Subsequently he restricts his focus through the property `drives`, specifically he restricts his focus to `European`. At that point he asks to change the entity type to `Cars`. This means that the entity type of the extension of the new state should be `Cars`, and the extension of the new state will contain *European cars which are driven by persons working at FORTH.* The property `drives` (actually its inverse direction), is now a possible facet of the current focus (and a condition is already active, based on the session of the user). Furthermore, the user can proceed and restrict his focus (*European cars which are driven by persons working at FORTH*) to those which are `ForSale`, and so on.

4.2 The Interaction Model for Fuzzy RDF

The general idea is that each transition of the model is now analyzed into $|\Phi|$ transitions, one for each $\varphi \in \Phi$, and each one is signified by its count (therefore now we will have $|\Phi|$ instead of 1 counts). To define these transitions we extend the previous definitions so that each of them takes an interval as additional parameter. Specifically, each *Restrict* takes as input an additional parameter φ, and the same for *Joins*, i.e.:

$$\begin{aligned} Restrict(E, c, \varphi) &= \{\ e \in E \mid e \in inst(c, \varphi)\} \\ Restrict(E, p : v, \varphi) &= \{\ e \in E \mid (e, p, v) \in inst(p, \varphi)\} \\ Restrict(E, p : vset, \varphi) &= \{\ e \in E \mid \exists\ v' \in vset \text{ and } \ (e, p, v') \in inst(p, \varphi)\} \\ Joins(E, p, \varphi) &= \{\ v \mid \exists\ e \in E \text{ and } \ (e, p, v) \in inst(p, \varphi)\} \end{aligned}$$

Regarding *class-based transitions*, it follows that for each tm in $TM_{cl}(s.e)$ we now have one $TM_{cl}(s.e, \varphi)$ for each $\varphi \in \Phi$, where:
$TM_{cl}(s.e, \varphi) = \{c \in C \mid Restrict(s.e, c, \varphi) \neq \emptyset\}$, and if the user clicks on a $c \in TM_{cl}(s.e, \varphi)$, then $s'.e = Restrict(s.e, c, \varphi)$.

Regarding *property-based transitions*, for each $p \in Props(s)$, the corresponding transition markers in plain RDF were defined by $Joins(s.e, p)$. Now, each element in $Joins(s.e, p)$ is analyzed to one $Joins(s.e, p, \varphi)$ for each $\varphi \in \Phi$. If the user clicks on a value v in $Joins(s.e, p, \varphi)$, then $s'.e = Restrict(s.e, p : v, \varphi)$. Regarding *presentation*, we do not show intervals, instead we show the corresponding count information. For example, for each $e \in E = \bigcup_{\varphi \in \Phi} TM_{cl}(s.e, \varphi)$ we show e once and its counts for each $\varphi \in \Phi$. Analogously for properties.

Let's now focus on *property paths*. For example consider two property instances pi_1 and pi_2 that form a path (e.g. `(cr2,inv(uses),Bob,0.2)` and `(Bob, worksAt,CSD,0.8)`), each associated with a fuzzy degree d_1 and d_2 respectively. The degree of path $pi_1 \cdot pi_2$ is $\min(d_1, d_2)$, in our case 0.2, since each path actually corresponds to a conjunction. This means that if the user's focus is cars and he wants to restrict it through the organization of the users of the cars, then the path $pi_1 \cdot pi_2$ will be taken into account for computing the count of the transition marker `CSD` whose interval encloses the degree $\min(d_1, d_2)$. To define this precisely, we first introduce some notations. Let $pp = p_1, \ldots, p_k$ be a property path. An *instance path of pp* is a sequence of the form $ip = (v_0, p_1, v_1) \cdot \ldots \cdot (v_{k-1}, p_k, v_k)$ where for all $1 \leq i \leq k$: $(v_{i-1}, p_i, v_i) \in \mathcal{C}(K)$. The *degree* of an instance path ip is defined as the *minimum* degree of its edges (property instance triples). The *degree of a path from o to o′* over pp, denoted as $degree(o, pp, o')$, is the *maximum* degree of all instance paths of pp between these two objects. We can now define *joins* and *restrictions* based on *fuzzy paths*:

$$Joins(E, pp, \varphi) = \{ \ v_k \mid \exists \ e \in E \text{ such that } \ degree(e, pp, v_k) \in \varphi \} \quad (6)$$

$$Restrict(E, pp : v_k, \varphi) = \{ \ e \in E \mid degree(e, pp, v_k) \in \varphi \} \quad (7)$$

Now we will analyze the algorithmic aspect of the above (since the previous two definitions were declarative). Consider a property path $pp = p_1 \cdot \ldots \cdot p_k$. The transition markers at each stage are defined as before, i.e. $M_i = Joins(M_{i-1}, p_i)$. For each individual element $e \in s.e$ we define the set of transition markers of level i (where $1 \leq i \leq k$) which are associated with it, as:

$$ETM_i(e) = \{m_i \in M_i \mid \exists \ s \in ETM_{i-1}(e) \ \text{ and } \ (s, p_i, m_i) \in inst(p_i)\} \quad (8)$$

assuming that $ETM_0(e) = \{e\}$.

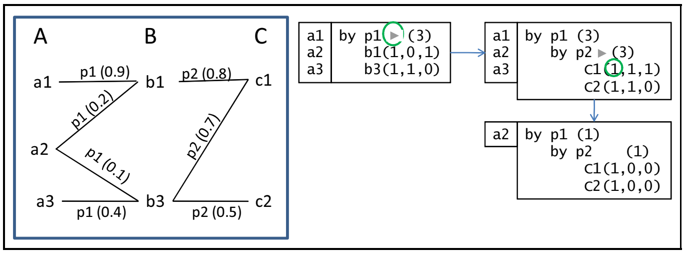


Fig. 5. Interaction over fuzzy paths

For example consider the case shown at Fig. 5. Let A be the extension of the current state ($M_0 = A = s.e$). For a path consisting only of one property p_1 we have that $M_1 = Joins(M_0, p_1) = \{b1, b3\}$, while for $p_1 \cdot p_2$ we have $M_2 = Joins(M_1, p_2) = \{c1, c2\}$. Now, for each element $e \in s.e$ we have the following sets of transition markers of level i:

level 0:	$ETM_0(a1) = \{a1\}$	$ETM_0(a2) = \{a2\}$	$ETM_0(a3) = \{a3\}$
level 1:	$ETM_1(a1) = \{b1\}$	$ETM_1(a2) = \{b1, b3\}$	$ETM_1(a3) = \{b3\}$
level 2:	$ETM_2(a1) = \{c1\}$	$ETM_2(a2) = \{c1, c2\}$	$ETM_2(a3) = \{c1, c2\}$

In addition, for each element $e \in s.e$ and transition marker $m_i \in ETM_i(e)$, we introduce a value denoted by $Deg(e, m_i)$, which is actually the degree of a path from e to m_i over pp (note that if pp is empty then we assume $Deg(e, e) = 1$). This value can be computed gradually (i.e. as the path gets expanded) as follows:

$$Deg(e, m_i) = \max_{m_{i-1} \in ETM_{i-1}(e)} \{\min(degree(m_{i-1}, p_i, m_i), Deg(e, m_{i-1}))\} \quad (9)$$

In our example we have: $Deg(a2, b1) = \max_{a \in ETM_0(a2)}\{\min(degree(a, p_1, b1), Deg(a2, a))\} = \min(degree(a2, p_1, b1), Deg(a2, a2)) = 0.2$. Analogously, $Deg(a2, b3) = 0.1$. Now, the degree of $a2$ to the transition marker $c1$ is computed as: $Deg(a2, c1) = \max_{b \in ETM_1(a2)}\{\min(degree(b, p_2, c1), Deg(a2, b))\} = \max\{\min(degree(b1, p_2, c1), Deg(a2, b1)), \min(degree(b3, p_2, c1), Deg(a2, b3))\} = \max\{\min(0.8, 0.2), \min(0.7, 0.1)\} = \max\{0.2, 0.1\} = 0.2$. Analogously, $Deg(a1, c1) = 0.8$ and $Deg(a3, c1) = 0.4$.

Finally, the *count* for each m_i of M_i that corresponds to φ is given by:

$$count(m_i, \varphi) = |\{e \in s.e \mid Deg(e, m_i) \in \varphi\}| \quad (10)$$

e.g. at Fig. 5 and for $\varphi = (0, 0.3]$, we have $count(c1, \varphi) = 1$. By clicking on the count $count(m_i, \varphi)$ the extension of the current state is restricted as follows $s'.e = \{e \in s.e \mid Deg(e, m_i) \in \varphi\}$.

4.3 Path Expansion and Cycles

Here, we examine how the interaction model for Fuzzy RDF behaves in case we have cycles at schema and instance level. At schema level, a property sequence may have the same starting and ending class forming a cycle (cyclic properties can be considered as a special case where the length of the sequence is 1). In the context of the proposed interaction model, when we have a cyclic schema path (e.g. $pp =$`inv(uses).worksAt.owns` as it is shown in Fig. 6) we may reach transitions markers which are also elements of the initial focus $s.e$, e.g. a transition marker m_i such that $m_i \in s.e$, and we may have to compute its $Deg(m_i, m_i)$. Consider the property path $p_1.p_2.p_3.p_4$ where $p_1 = inv(uses)$, $p_2 = knows$, $p_3 = worksAt$, $p_4 = owns$. If $cr1$ belongs to $s.e$, then $ETM_0(cr1) = \{cr1\}$, $ETM_1(cr1) = \{Alice\}$, $ETM_2(cr1) = \{Bob\}$, $ETM_3(cr1) = \{CSD\}$ and $ETM_4(cr1) = \{cr1\}$. To compute the appropriate count information of the transition marker *cr1*, it is enough to compute $Deg(cr1, cr1)$ according to Eq. 9, as the path $pp = p_1.p_2.p_3.p_4$ is not empty. The point is that the proposed model can handle cycles at schema level without requiring any further configuration.

Now consider a user who wants to restrict the initial focus set, being a set of persons, through other persons connected with them through the property *knows* at depth *m*. For this reason the user expands the property *knows m* times and then selects a person. However, at some point we may want to stop

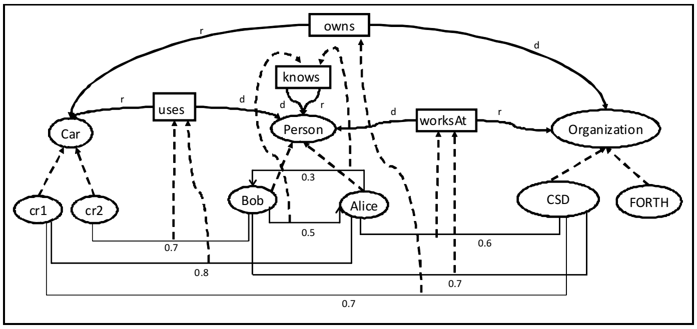
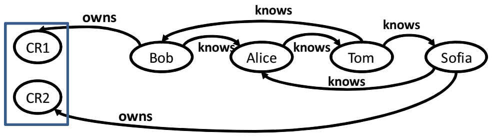


Fig. 6. A Fuzzy RDF graph with cycles at schema and instance level



TMs for inv(owns).knows^i		ipath from cr1	ipath from cr2
i=1	A(2)	B.**A**	S.**A**
i=2	T(2)	B.A.**T**	S.A.**T**
i=3	S(2), B(2)	B.A.T.**S**, B.A.T.**B**	S.A.T.**S**, S.A.T.**B**
i=4	A(2)	B.A.T.S.**A**, B.A.T.B.**A**	S.A.T.S.**A**, S.A.T.B.**A**
i=5	T(2)	B.A.T.S.A.**T**, B.A.T.B.A.**T**	S.A.T.S.A.**T**, S.A.T.B.A.**T**

Fig. 7. Instance cycles example

suggesting path expansions, in order to avoid prompting to the user the same set of transition markers (which may periodically restrict the initial focus in the same way). For example, in Fig. 7 we can see that when the property `knows` is expanded over 3 times, the transition markers and their count info is periodically being repeated[3]. We propose adopting the following policy: *stop path expansion when each object of the s.e has been made accessible (i.e. restrictable) through all transition markers that are possible.* Below we explain how we can compute the max number of expansion steps. Let $\Gamma = (N, R)$ be a directed graph. A *path* from n_1 to n_{k+1}, is a sequence of edges of the form $(n_1, n_2) \dots (n_k, n_{k+1})$ where $(n_i, n_{i+1}) \in R$, and $i \neq j$ implies that $n_i \neq n_j$. The length of such a path is k, and we shall write $n_1 \overset{k}{\rightsquigarrow} n_{k+1}$ to denote that there exists a path of length k from n_1 to n_{k+1}. We shall also write $n \overset{*}{\rightsquigarrow} n'$ to denote that exists one or more paths from n to n'. Now we define the distance from n to n' as the length of the *shortest* path from n to n', i.e. $Dist(n \rightarrow n') = \min\{\ k \mid n \overset{k}{\rightsquigarrow} n'\}$. Given two subsets A and B of N (i.e. $A, B \subseteq N$), we define the distance from A to B as the *maximum* distance between any pair of nodes form these sets, i.e. $Dist(A, B) = \max\{\ Dist(a \rightarrow b) \mid a \in A, b \in B\}$.

Returning to our problem N is the set of all nodes of the RDF graph. For a property $p \in Pr$ we can define the edges $R_p = \{(a, b) \mid (a, p, b) \in \mathcal{C}(K)\}$. We can

[3] Only the first letter of a name is shown and paths over `knows` are depicted as sequences of such letters.

now define the reachable nodes from a node n (through p property instances) as $Reachable_p(n) = \{\ n' \mid n \overset{*}{\rightsquigarrow}_p n'\}$, where the meaning of the subscript p is that paths are formed from elements of R_p. Being at a state s, the maximum number of path expansion steps (for property p) that is required are:

$$MaxExpansionSteps(s,p) = Dist(s.e, \bigcup_{n \in s.e} Reachable_p(n))$$

With this number of steps it is guaranteed that each object of $s.e$ has been made accessible (restrictable) through all tms (transition markers) which are possible. The proof is trivial: the path starting from an object o with length bigger than $MaxExpansionSteps(s,p)$ will not encounter a tm that has not already been reached.

Now consider path expansions over different properties (i.e. `inv(owns).knows.knows`). In such cases we would like to to identify the maximum expansion steps for each p that is used in the expansion, or the maximum expansion steps in general. Let $pset$ be a set of properties (i.e. $pset \subseteq Pr$). We can define the edges by considering all properties in $pset$, i.e. $R_{pset} = \{(a,b) \mid p \in pset,\ (a,p,b) \in \mathcal{C}(K)\}$. Now we can define $Reachable_{pset}(n) = \{\ n' \mid n \overset{*}{\rightsquigarrow}_{pset} n'\}$, where the subscript $pset$ means that paths consist of edges in R_{pset}. The set $Reachable_{pset}(n)$ is the set of all tms through which n is accessible. Therefore the tms of all objects in $s.e$, which are accessible through paths using property instances in $pset$, are given by $\bigcup_{n \in s.e} Reachable_{pset}(n)$. Being at a state s, the maximum number of path expansion steps (using properties from $pset$) that is required is:

$$MaxExpansionSteps(s,pset) = Dist(s.e, \bigcup_{n \in s.e} Reachable_{pset}(n))$$

5 Query Language Issues

We have defined the interaction model using only extensions (not intentions), since the expression of the intention depends on the Query Language (QL), or the abstraction of the QL that one adopts. However the underlying information source may be accessible through a QL. Table 1 shows the notation we have used for defining RDF browsing, and their expression in SPARQL. In this description we consider that the extension of the current state is stored in a class with name `ns:temp`[4]. Furthermore we assume that the closure of the KB is stored. However, we should note that *Virtuoso* [3], supports an extended SPARQL version with *subclassOf* and *subproperty* inference at query level. This means that triples entailed by subclass or subproperty statements are not physically stored, but they are added to the result set during query answering[5]. This means that the SPARQL expressions of Table 1 would not require another change.

[4] Instead of `ns:temp` we could have a set of SPARQL graph patterns. For reasons of space, we do not describe the query construction method.

[5] Similar in spirit with the online approach [5] to query the Web of Linked Data by traversing RDF links during run-time.

Table 1. SPARQL-expression of Notations for RDF Browsing

Notation	Expression in SPARQL
$Restrict(E, p : vset)$, $vset = \{v_1, ..., v_k\}$	`select ?x where { ?x rdf:type ns:temp; ns:p ?V.` `Filter (?V = ns:v_1 ||...|| ?V = ns:v_k)}`
$Restrict(E, c)$	`select ?x where { ?x rdf:type ns:temp; rdf:type ns:c.}`
$Joins(E, p)$, where $E = \{e_1, ..., e_k\}$	`select Distinct ?v where { ?x ns:p ?v.` `Filter (?x = ns:e_1 ||...|| ?x = ns:e_k)}`
$TM_{cl}(s.e)$ and counts	`select Distinct ?c count(*) where{?x rdf:type ?c; rdf:type ns:temp.}` `group by ?c`
$Props(s)$	`select Distinct ?p where{ {?x rdf:type ns:temp; ?p ?v.}` `UNION {?m rdf:type ns:temp. ?n ?p ?m. }}`
$Joins(s.e, p)$ and counts	`select Distinct ?v count(*) where{ ?x rdf:type ns:temp; ns:p ?v.}` `groupby ?v`

Regarding QLs for Fuzzy RDF, there is not any standardized (or widely adopted) extension of SPARQL. Some recently proposed deductive systems, e.g. [16,15], support fuzzy answering over unions of conjunctive queries, by computing the closure of a Fuzzy RDF graph (i.e. $degree(o, type, c)$ is computed as we have defined it, however $degree(o, p, o')$ is not directly supported), storing it into a relational DB, and then using internally SQL queries. For instance, [16] uses MonetDB with the following schema: `type(subject, object, degree)`, `subclassOf(subject, object, degree)`, `subpropertyOf(subject, object, degree)`, and a table $prop_i$`(subject, object, degree)` for every distinct property p_i. Table 2 shows directly the SQL queries that are needed by our interaction model for Fuzzy RDF (again E could also be defined through another query).

Table 2. SQL-expression of Notations for Fuzzy RDF Browsing

Notation	Expression in SQL
$Restrict(E, c, \varphi)$	`select subject from type` `where object='c' and subject in E` `and degree>phi.low and degree<=phi.up`
$Restrict(E, p : vset, \varphi)$	`select subject from p` `where object in VSET and subject in E` `and degree>phi.low and degree<=phi.up`
$Joins(E, p_i)$	`select object from p_i  where subject in E`
$TM_{cl}(s, \varphi)$ and counts	`select object, sum(case when degree>phi.lower` `and degree <= phi.upper then 1 else 0 end),` `from type` `where subject in s.e  group by object`
$ETM(e, M_i, Prev)$	`select object from p_i` `where subject in Prev and object in M_i`
$Deg_{MIN}(e, subj, m_i, d)$, where $subj \in ETM_{i-1}(e)$ and $d = Deg(e, subj)$	`select` `case when degree > d then d else degree end as DEG_MIN` `from p where subject='subj' and object='m_i'`

Regarding property paths, at each step we can compute $M_i = Joins(M_{i-1}, p_i)$ with a single SQL query (as shown in Table 2). The difference of fuzzy paths vs non fuzzy paths, is that for moving from a stage i of the path to a stage $i+1$, we have for each $e \in s.e$ to keep (a) $ETM_i(e)$, and (b) $Deg(e, m_i)$ for each $m_i \in ETM_i(e) \subseteq M_i$. To compute $ETM_i(e)$ we can use a query of the form $ETM(e, M_i, Prev)$ (shown in Table 2) where $Prev = ETM_{i-1}(e)$. To compute $Deg(e, m_i)$ we can use a query of the form $Deg_{MIN}(e, subj, m_i, d)$ for every $subj \in ETM_{i-1}(e)$ (and accordingly $d = Deg(e, subj)$) and then get the max.

6 Conclusion

We proposed a session-based interaction model for exploring Fuzzy RDF KBs in a simple and intuitive manner. To exploit fuzzy degrees the model supports interval-based transition markers and this (exponentially) increases the discrimination power of the interaction. Roughly, for each condition of a (formulated on the fly) query, we have $|\Phi|$ refinements of that condition. This means that for states corresponding to queries with k conditions we now have $|\Phi|^k$ more states. This increase does not affect the friendliness of the interaction since only states leading to non-empty results are given. We also analyzed the query language requirements for realizing this model on top of the query layer. We do not include experimental measurements, since this is not the focus of this paper (and for reasons of space), however we should mention that to compute the class-based transition markers, and their fuzzy counts, for a dataset with 10^7 instances can take up to 3 secs for the case of $|\Phi| = 5$ in MonetDB[6]. Directions for further research regard ranking methods for the transition markers. For instance, fuzzy degrees can be exploited for clustering transition markers, or for ranking them through more refined methods than those proposed for plain RDF (e.g. [11]).

References

1. Allard, P., Ferré, S.: Dynamic Taxonomies for the Semantic Web. In: Procs. of the 19th Intern. Conf. on Database and Expert Systems Application, DEXA (2008)
2. Athanasis, N., Christophides, V., Kotzinos, D.: Generating On the Fly Queries for the Semantic Web: The ICS-FORTH Graphical RQL Interface (GRQL). In: McIlraith, S.A., Plexousakis, D., van Harmelen, F. (eds.) ISWC 2004. LNCS, vol. 3298, pp. 486–501. Springer, Heidelberg (2004)
3. Erling, O., Mikhailov, I.: RDF Support in the Virtuoso DBMS. In: Procs. of 1st Conf. on Social Semantic Web (2007)
4. Harth, A.: VisiNav: Visual web data search and navigation. In: Bhowmick, S.S., Küng, J., Wagner, R. (eds.) DEXA 2009. LNCS, vol. 5690, pp. 214–228. Springer, Heidelberg (2009)
5. 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)
6. Holi, M., Hyvönen, E.: Fuzzy view-based semantic search. In: Mizoguchi, R., Shi, Z.-Z., Giunchiglia, F. (eds.) ASWC 2006. LNCS, vol. 4185, pp. 351–365. Springer, Heidelberg (2006)
7. Kohlschütter, C.: Using link analysis to identify aspects in faceted web search. In: SIGIR 2006 Workshop on Faceted Search, pp. 55–59 (2006)
8. Lu, J., Zhu, Y., Zeng, X., Koehl, L., Ma, J., Zhang, G.: A fuzzy decision support system for garment new product development. In: Australasian Conf. on Artificial Intelligence, pp. 532–543 (2008)

[6] More in the extended version of this paper (on preparation).

9. Mäkelä, E., Hyvönen, E., Saarela, S.: Ontogator — A Semantic View-Based Search Engine Service for Web Applications. 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. 847–860. Springer, Heidelberg (2006)
10. Mazzieri, M.: A Fuzzy RDF Semantics to Represent Trust Metadata. In: 1st Workshop on Semantic Web Applications and Perspectives, SWAP 2004 (2004)
11. Oren, E., Delbru, R., Decker, S.: Extending Faceted Navigation for RDF Data. 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. 559–572. Springer, Heidelberg (2006)
12. Papadakos, P., Kopidaki, S., Armenatzoglou, N., Tzitzikas, Y.: On exploiting static and dynamically mined metadata for exploratory web searching. Knowledge and Information Systems (accepted for publication in 2011) issn 0219-1377
13. Sacco, G.M.: e-RARE: Interactive Diagnostic Assistance for Rare Diseases through Dynamic Taxonomies. In: DEXA Workshops (2008)
14. Stefaner, M., Urban, T., Seefelder, M.: Elastic lists for facet browsing and resource analysis in the enterprise. In: Bhowmick, S.S., Küng, J., Wagner, R. (eds.) DEXA 2008. LNCS, vol. 5181, pp. 397–401. Springer, Heidelberg (2008)
15. Straccia, U., Lopes, N., Lukacsy, G., Polleres, A.: A general framework for representing and reasoning with annotated semantic web data. In: Twenty-Fourth AAAI Conf. on Artificial Intelligence, AAAI 2010 (2010)
16. Straccia, U.: A Minimal Deductive System for General Fuzzy RDF. In: Polleres, A., Swift, T. (eds.) RR 2009. LNCS, vol. 5837, pp. 166–181. Springer, Heidelberg (2009)
17. Tilsner, M., Hoeber, O., Fiech, A.: Cubansea: Cluster-based visualization of search results. In: Procs of the Inter. Joint Conf. on Web Intelligence and Intelligent Agent Technology, WI-IAT 2009 (2009)
18. Udrea, O., Recupero, D.R., Subrahmanian, V.S.: Annotated RDF. ACM Trans. Comput. Logic 11(2) (2010)
19. Wu, J.-K., Desai Narasimhalu, A., Mehtre, B.M., Lam, C.-P., Gao, Y.J.: Core: A content-based retrieval engine for multimedia information systems. Multimedia Syst. 3(1), 25–41 (1995)
20. Yee, K.-P., Swearingen, K., Li, K., Hearst, M.: Faceted metadata for image search and browsing. In: Procs. of the SIGCHI Conf. on Human factors in Computing Systems (2003)

A Structured Semantic Query Interface for Reasoning-Based Search and Retrieval

Dimitrios A. Koutsomitropoulos[1], Ricardo Borillo Domenech[2], and Georgia D. Solomou[1]

[1] High Performance Information Systems Laboratory (HPCLab), Computer Engineering and Informatics Dpt., School of Engineering, University of Patras, Buidling B, 26500 Patras-Rio, Greece
[2] Servicio de Informática, Universitat Jaume I, Rectorado, 12071, Castellón, Spain
kotsomit@hpclab.ceid.upatras.gr, borillo@uji.es, solomou@hpclab.ceid.upatras.gr

Abstract. Information and knowledge retrieval are today some of the main assets of the Semantic Web. However, a notable immaturity still exists, as to what tools, methods and standards may be used to effectively achieve these goals. No matter what approach is actually followed, querying Semantic Web information often requires deep knowledge of the ontological syntax, the querying protocol and the knowledge base structure as well as a careful elaboration of the query itself, in order to extract the desired results. In this paper, we propose a structured semantic query interface that helps to construct and submit entailment-based queries in an intuitive way. It is designed so as to capture the meaning of the intended user query, regardless of the formalism actually being used, and to transparently formulate one in reasoner-compatible format. This interface has been deployed on top of the semantic search prototype of the DSpace digital repository system.

Keywords: Semantic Web, queries, ontologies, entailment, guided input.

1 Introduction

The growing availability of semantic information in today's Web makes ontology-based querying mechanisms necessary. Europeana for example counts over 10M of semantic objects corresponding to heritage and collective memory resources [14]. And this currently forms only the tip of the iceberg: Vast amounts of Linked Data exist and continuously emerge out of DBpedia, social applications, open government data and other sources.

However, querying the Semantic Web is not a straightforward task, especially in case of expressive ontology languages, like OWL and OWL 2 where inference holds a key part. In addition to the current lack of protocols and standards for efficiently

© Springer-Verlag Berlin Heidelberg 2011

searching through ontological information, one has to cope with the added complexity Semantic Web queries inherently bear with:

- *A priori ontological knowledge*: In order to formulate an expressive query and to bound results, the user needs to know in advance class names, properties and individuals that consist the ontology's contents. Alternatively, suitable mechanisms are necessary to expose this information to the user.
- *Expert syntaxes,* like Description Logics (DLs), SPARQL, Manchester Syntax, that are usually difficult to read and comprehend, let alone to compose from scratch for non-expert users.
- *Inherent complexity*: The added-value of ontological, entailment-based querying does not surface unless a suitable query expression is as elaborate as possible. In order to surpass the level of relational queries, one needs to delve into complex combinations of classes, properties and restrictions, thus formulating expressive conjunctive queries, impossible to express or answer using relational techniques [9].

Finally, *NLP approaches* are not always a sound solution: While they can produce meaningful results in a number of cases [8], there is virtually no guarantee that the intended user query will actually be captured. Parsing a free text sentence may or may not correspond to a successful query expression or to the one that the user would have meant to.

Therefore in this paper we propose a *structured* querying mechanism and interface that helps to construct and submit entailment-based queries to web ontology documents. The main idea is to aid the user in breaking down his intended query expression into several *atoms*. These atoms are then combined to form allowed expressions in Manchester Syntax, as the closest to our purposes regarding user-friendliness. At the same time, the interface tries to be as intuitive as possible by automatically disallowing (graying out) nonsensical combinations (for example, select a restriction without selecting a property first), offering dynamic auto-complete choices and classify them as per class (type) or relation, disclosing namespace prefixes when possible, marking the various fields with NL-like labels and presenting results based on their class or type.

Based on this idea we have developed a prototype application as an add-on to the DSpace digital repository system, latest version (1.6.2 and 1.7.0)[1]. This work builds upon and evolves earlier efforts for creating a semantic search service for DSpace [10]. The novel semantic search interface is backed up by a new *DSpace Semantic API* that supports a pluggable design for reasoners as well as OWL 2.0 and the newest OWL API v.3. Most importantly, our Semantic API is designed along the same principles as its predecessor, i.e. to remain independent of the DSpace business logic and to be agnostic to the rest of the user interface or even the underlying ontology.

In the following we first review current approaches for querying the Semantic Web and point out our decisions for the interface design (Section 2). Then we describe the design and architecture of the DSpace Semantic API, which the querying services are

[1] `http://www.dspace.org/`

based on (Section 3). Section 4 describes the user-perceived functionality of our interface and presents some indicative examples. Finally, section 5 and 6 summarize our conclusions and future work.

Our prototype is openly available at: http://apollo.hpclab.ceid.upatras.gr:8000/jspui16/semantic-search. Source code is maintained as a Google Code project[2] where instructions and latest developments can be found.

2 Background

As long as there is not yet a standard query language specifically for OWL ontologies, a search mechanism that intends to utilize a formal query language has to choose among either a DL-based or a RDF-based approach. The former category is more closely related to rule languages and logics. The latter includes SQL-like languages, aiming at retrieving information from RDF documents. No matter what approach is actually followed, Semantic Web query effectiveness highly depends on the mechanisms employed to actually construct the query, as discussed in the previous section.

2.1 Syntaxes for OWL Query Languages

Some known languages for querying RDF data are SPARQL [15], SeRQL[1], RDQL[16] and more. But SPARQL is the one that has been recognized as the de facto standard for the Semantic Web. Because SPARQL has been mainly designed for querying RDF documents, its semantics are based on the notion of RDF graphs and thus it has no native understanding of OWL vocabulary.

Even when a bridging axiomatization is offered (like for example with SPARQL-DL [18]), OWL-oriented queries in SPARQL can become extremely verbose, especially in case complex OWL expressions are involved. To this end, some SPARQL variants have been recently proposed, such as SPARQLAS[3] and Terp [17], bearing a more OWL-friendly profile: they both intend to facilitate those who are not familiar with SPARQL to write queries in this language, by allowing the mix with OWL syntactic idioms, like OWL functional syntax and OWL Manchester Syntax, respectively.

Nevertheless, DL-based query languages, compared to the RDF-based ones, have more well-defined semantics w.r.t. OWL since they were designed exactly for querying OWL (and OWL 2) documents. For example nRQL [4], OWL-QL [3] and OWLlink's ASK protocol [11] fall in this category.

Provided that querying an ontology is often about finding individuals or instances, another simple yet powerful approach is to directly employ DL syntax to write these queries. For example, the query tab of Protégé 4[4] follows this practice. It uses the Manchester syntax [6], which is a "less logician like" and more user-friendly syntax for writing OWL class expressions.

[2] `http://code.google.com/p/dspace-semantic-search/`
[3] `http://code.google.com/p/twouse/wiki/SPARQLAS`
[4] `http://protegewiki.stanford.edu/wiki/DLQueryTab`

When a user poses a query, a parser maps the given expression into a concept expression (class). Consequently, all instances classified by the reasoner under this particular concept are retrieved. In this sense, Manchester syntax can be used as an entailment-based querying language for OWL documents and this is the approach we follow in our implementation.

2.2 Query Formulation

The existing approaches for performing search on the Semantic Web can be roughly divided in two categories [2]: those using structured query languages, like the ones described previously, and those expressing natural language or keyword-based queries. The first category includes systems that use a formal language for evaluating queries. The latter category is comprised of applications that accept either a whole phrase (expressed in natural language) or simple keywords as queries. The NL-based approaches usually require an additional reformulation step where posed queries are translated into class expressions or triples, depending on the target language. Such a system is PowerAqua [12], where a user query is translated from natural language into a structured format. Similarly, in keyword-based systems, keywords are matched to parts of an RDF graph or to ontology elements and then evaluated against the knowledge base. QUICK [20] belongs to this category and is based on the idea of assigning keywords to ontology concepts.

The parsing and reformulation process that is necessary in NL-based and keyword based approaches restricts systems functionality, as it involves further query analyzing, and sometimes makes handling complex requests difficult or even impossible. On the other hand, the use of formal structured languages in query interfaces assumes that users have at least a basic understanding of the language's syntax as well as of the underlying ontology's structure. This requirement leads to more expert-user oriented applications, not suitable for common users who are not familiar with the logic-based Semantic Web.

A way to bridge the gap between the complexity of the target query language and end users is to develop a guided input query interface. This is a practice followed along several years, by applications querying database systems, in order to facilitate the formulation of more complex SQL requests. For example, an advanced search facility in a digital library system, like DSpace, utilizes drop down menus with Boolean operators so as to help users in setting restrictions when searching the system's database. When searching knowledge bases, though, where ontologies are involved, things become much more complicated.

Several semantic search systems that guide users in structuring their requests have been proposed in the literature, but this is mostly about systems that use SPARQL and SPARQL-like languages. In addition, they are usually focused on graphical or visual techniques, like NITELIGHT [19] and Konduit [13]. To the best of our knowledge no other system using DL-based query languages exists, that follows the idea of controlled input forms for the structured formulation of semantic queries.

3 The DSpace Semantic API

In this section we focus on the design and implementation of the semantic search service, which has been developed as an add-on to the new DSpace 1.7. We describe the main components of the Semantic API and then we point out its interaction with inference engines in order to support entailment-based queries. First however we briefly introduce the DSpace ontology model which acts as the underlying knowledge base for queries.

3.1 The DSpace Ontology

The first step in developing any semantic search service is to identify or construct the target knowledge base or ontology, which queries will actually be performed against. In our case, we construct the DSpace ontology on-the-fly, following a sophisticated procedure fully described in [9] and based on the interoperable system's mechanisms for exporting resources' metadata through OAI-PMH.

DSpace metadata follow the Dublin Core (DC) specification by default, while it is possible to import and use other metadata schemata as well. In our particular implementation, we have also enhanced the system's metadata schema with learning object (LOM) metadata, specifically tailored for its usage as an institutional repository (http://repository.upatras.gr/dspace/). The resulting ontology comprises of axioms and facts about repository items and is expressed in OWL 2 (Fig. 1).

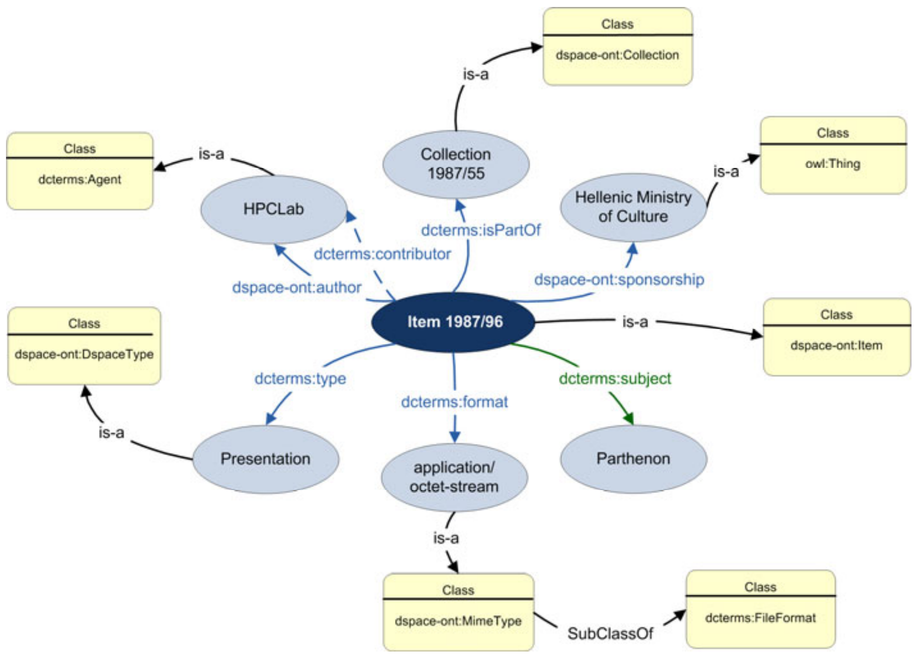


Fig. 1. An example instance of the DSpace ontology

3.2 Design and Architecture

The semantic search service uses several APIs to perform search and inference against the ontology. The OWL API [5] is used as the basis for ontology manipulation and interaction with reasoners. Compared to our previous efforts [10] we have now migrated to the latest OWL API v. 3.1.0 to support proper handling of OWL 2 idioms as well as to better interface various reasoners. For the latter, we have also upgraded to FaCT++ v. 1.5.0[5] and added the ability to "hot-swap" between reasoners dynamically, adding support also for Pellet[6].

Figure 2 depicts the different components of the semantic search service in relation to the DSpace infrastructure.

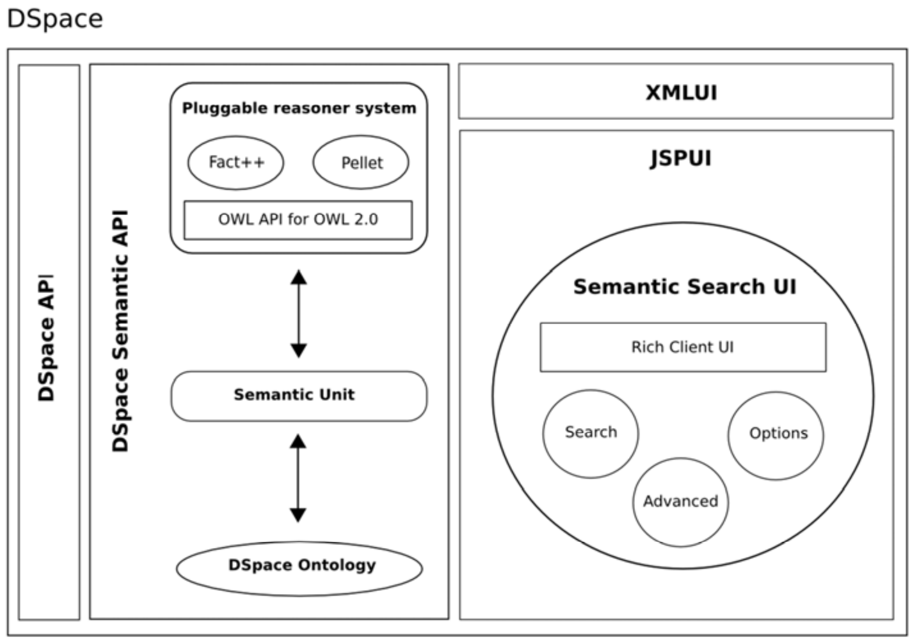


Fig. 2. The architecture of the semantic search service for DSpace

All these components are part of the DSpace Semantic API. The DSpace Semantic API is defined at the same level as the DSpace API. This new API can be used in the rest of DSpace modules without problems, like the JSP user interface (JSPUI), XMLUI, REST API or LNI. In our implementation, the DSpace Semantic API interacts with the JSPUI module by means of a new user interface for querying DSpace digital objects using an ontology (see Section 4).

The *Semantic Unit* is the core component and the mediator between all the facts and relations defined in the DSpace ontology and the inference engine (Fact++, Pellet or another OWL API-compliant reasoner).

[5] `http://code.google.com/p/factplusplus/`
[6] `http://clarkparsia.com/pellet/`

3.3 The Semantic Unit

When the implementation of the semantic layer began, a core piece with all the basic and necessary resources to execute semantic queries was created. This main unit was called *Semantic Unit* and was designed as a singleton class to be available to the entire system and initialized at system startup with the DSpace ontology and other default values.

This unit is responsible for the following topics:

- *The OWL ontology manager*. An `OWLOntologyManager` manages a set of ontologies. It is the main point for creating, loading and accessing ontologies. An `OWLOntologyManager` also manages the mapping between an ontology and its ontology document.
- *The OWL ontology* itself.
- *The imports closure*. This is just the union or aggregation of the imported ontology documents, referenced by the `owl:imports` directive.
- *The short form provider*. In OWL, entities such as classes, properties and individuals are named using URIs. Since URIs can be long and not particularly readable, "short forms" of these URIs are often used for presentation in tools such as editors and end user applications. Some basic implementations of short form providers are:
  - `SimpleShortFormProvider`. Generates short forms directly from URIs. In general, if the fragment of a URI is available (the part of the URI following the #) then this will be used for the short form.
  - `QnameShortFormProvider`. Generates short forms that look like *QNames*. For example, `owl:Thing`, `pizza:MarghertiaPizza`.
- *The reasoner* used.

The Semantic Unit is also a registry for caching purposes. This allows to reuse the loaded ontologies and short form providers, avoiding reload and parsing of the whole ontology definition. As a result, when a user loads a new ontology, this is loaded and stored only once by the Semantic Unit in an internal registry. When another user asks for the same ontology, no re-parsing is needed, and the ontology is served from the registry.

This functionality is used around the system to perform some basic interactions. The Semantic Unit is used in the main module to issue queries with the values provided by the end user. Additionally, the Semantic Unit is also used in the construction of the search results page, when the user asks for a detailed view of a record and the system performs some inference against the ontology. Summarizing, the Semantic Unit will be used in every situation where the system needs to perform any operation against the loaded ontology.

3.4 Pluggable Reasoner Design

One of the design principles of the semantic search service was openness and support for different reasoners and ontologies. In this way, a proper design on source code is also needed.

To allow the Semantic API to load different reasoners dynamically, a little use of reflection and a general interface definition was used:

```
public interface OWLReasonerFactory
{
    public OWLReasoner getReasoner(OWLOntology
ontology);
}
```

This is the main interface that is required to be implemented for every reasoner that we want to incorporate to our semantic search service. This is only a factory method implementation to create the proper instance of the reasoner. That was needed because different reasoners have their own API for creating instances. Once a reasoner instance is generated, no customization is needed because of the fact that all reasoners implement the `OWLReasoner` OWL API interface.

This can be accomplished by using Java Reflection and by relying on system configuration to determine what the correct reasoner that needs to be loaded is:

```
SupportedReasoner supportedReasoner =
SupportedReasoner.PELLET;
OWLReasonerFactory owlReasonerFactory =
(OWLReasonerFactory) Class.forName(
supportedReasoner.toString()).newInstance();
OWLReasoner reasoner =
owlReasonerFactory.getReasoner(ontology);
```

All the supported reasoners are defined in a Java enumeration class:

```
public enum SupportedReasoner
{
FACTPLUSPLUS("gr.upatras.ceid.hpclab.reasoner.OWLReason
erFactoryFactPlusPlusImpl"),
PELLET("gr.upatras.ceid.hpclab.reasoner.OWLReasonerPell
etImpl");
    private String classImpl;
    SupportedReasoner(String value)
    {
        classImpl = value;
    }

    public String toString()
    {
        return classImpl;
    }
}
```

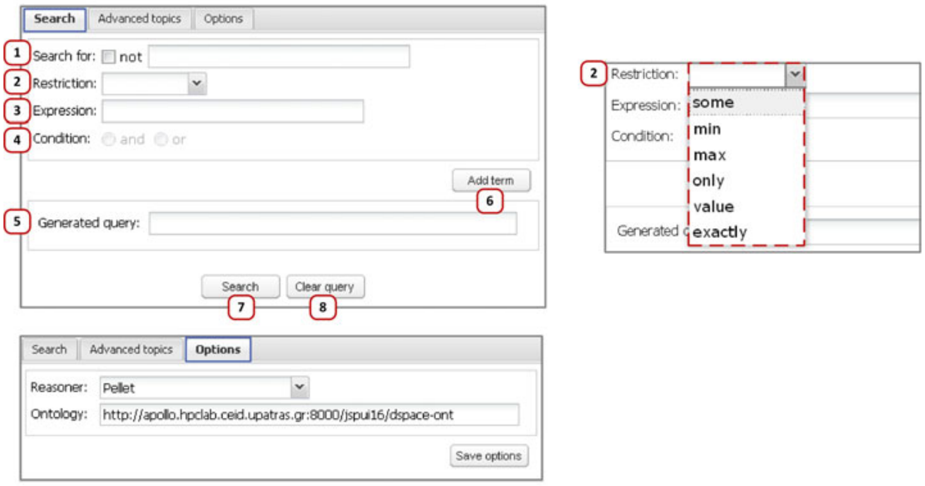
Following these guidelines two initial implementations called `OWLReasoner FactoryFactPlusPlusImpl` for Fact++ and `OWLReasonerPelletImpl` for Pellet support have been added.

4 Functionality and Examples

In this section we describe how our semantic search service interacts with users, guiding them smoothly in the construction of correct and accurate queries. First, a detailed description of the interface building components is given; then, we present how users can take advantage of this interface, by showing some indicative examples.

4.1 The Interface

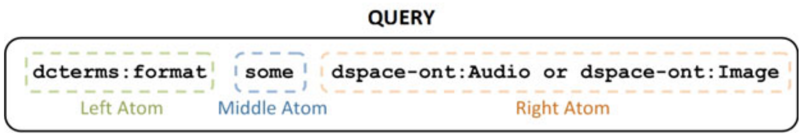
When the semantic search interface is loaded, one can distinguish among three separate tabs: *Search* (default), *Advanced topics* and *Options*.



Reasoner: Pellet
Ontology: http://apollo.hpclab.ceid.upatras.gr:8000/jspui16/dspace-ont

Fig. 3. *Search* and *Option* tab of the semantic search interface

The *Search* form contains all necessary elements for guiding users in building queries in Manchester syntax as intuitively as possible. Each component in this form corresponds to a certain building atom of the query expression. Their functionality is described in detail later in this section.



```
dcterms:format some dspace-ont:Audio or dspace-ont:Image
```

Fig. 4. The building atoms of a query expression in Manchester syntax

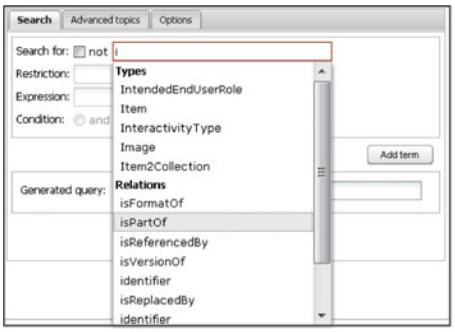
The *Advanced topics* tab is currently inactive and is reserved for future extensions of the system, like for example the support of other query syntaxes, such as SPARQL.

The *Options* tab includes options that allow users to change the ontology against which they perform their search, as well as the underlying reasoner (currently, Pellet or FaCT++). For altering the knowledge base, we only need to supply a valid URL of an OWL/OWL 2-compliant ontology. In addition, the user can switch between reasoners dynamically (Section 3.4).

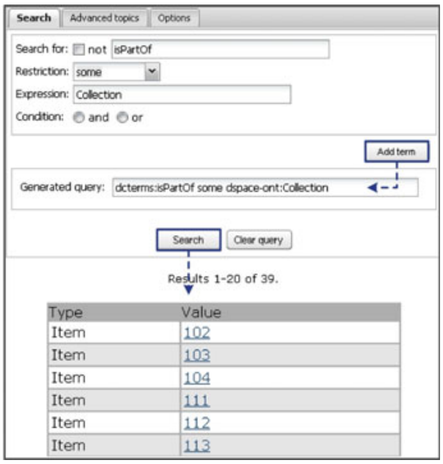
Next we describe the various elements of the *Search* tab, according to their number in Fig. 3. Based on Manchester syntax's primitives for formulating an expression [7] and depending on the values entered by the user in each preceding field, subsequent fields are enabled or disabled accordingly. Figure 4 depicts the three main atoms of such a query expression. What is more, an auto-complete mechanism is enabled where necessary, for guiding users in supplementing information.

1. **Search for:** It corresponds to the outmost left (first) atom of a Manchester syntax expression. This can be either a property name or a class name. An auto-complete mechanism is triggered as soon as a word starts being typed, suggesting names for classes and properties that exist in the loaded ontology. For users' convenience, suggested values have been grouped under the title *Types* (for classes) and *Relations* (for properties) (left part of Fig. 5). The check box is used for declaring the negation of the class expression that starts being constructed. For simplicity, all prefixes are kept hidden from users and the system is responsible for adding them automatically, during the query generation process. The following two fields are not activated, unless a property name has been selected in this step.
2. **Restriction:** This represents the middle atom of the expression. Provided that a property is entered in the previous field, a number-, value- or existential restriction should now be set. Hence, the 'Restriction' drop down menu becomes active, containing respective Manchester syntax keywords.
3. **Expression:** This is a free-text field where the user can supply a single class name or expression (quantification), an individual (value restriction) or a number, optionally followed by a class (cardinality restriction). An auto-complete facility is provided for class names. This forms the outmost right (last) atom of the query expression.
4. **Condition:** From now on, the user can recursively construct more class expressions, combining them in conjuctions (*and*) or disjunctions (*or*). Consequently an appropriate *Condition* should be set for expressing the type of logical connection.
5. **Generated Query:** This field gradually collects the various user selections and inputs, ultimately containing the final query expression. It is worth noting that this is an editable text box, meaning that expert users can always bypass the construction mechanism and use it directly for typing their query.
6. **Add term:** Adds a completed expression to the *Generated Query* field. This also checks if the expression to be added is valid, and pops an error message otherwise.
7. **Search:** When pressed, evaluates the query expression as it appears in the *Generated Query* field. It also clears all other fields, thus giving the user the opportunity to further refine his initial query.
8. **Clear query:** Clears the form and makes it ready to accept new values.

Once the query has been evaluated, obtained results appear right below the search form. They are organized in the form of a two-column table, as depicted in Fig. 5: *Value* contains the retrieved entities, whereas *Type* indicates at least one of the classes



Search for: not i
Restriction:
Expression:
Condition: and
Generated query:



Search for: not isPartOf
Restriction: some
Expression: Collection
Condition: and or
Generated query: dcterms:isPartOf some dspace-ont:Collection

Results 1-20 of 39.

Type	Value
Item	102
Item	103
Item	104
Item	111
Item	112
Item	113

Fig. 5. The auto-complete mechanism and the results table in the semantic search interface

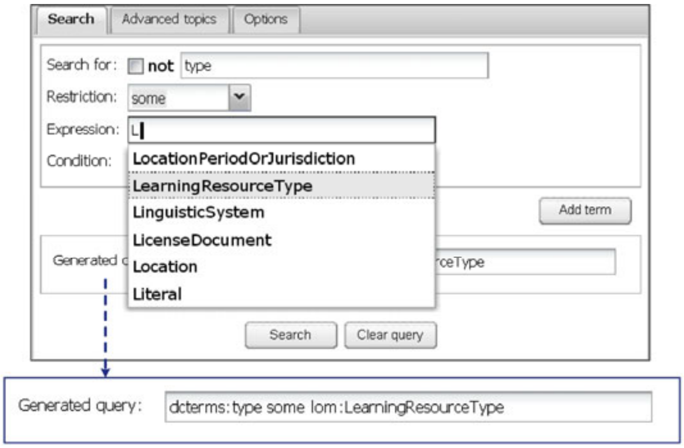
to which each entity belongs, thus providing users with a hint about their type. All retrieved entities are clickable and when selected, a separate page is loaded, containing the complete ontological information about the clicked entity. More details about this page and its elements can be found in [10].

4.2 Example Queries

First we show how a relatively simple class expression can be built through the interface. In particular, we want to retrieve all DSpace entities that have a type of `lom:LearningResourceType`. This corresponds to the following expression:

`dcterms:type some lom:LearningResourceType`

The way it is constructed through the semantic search interface is shown in Fig. 6.



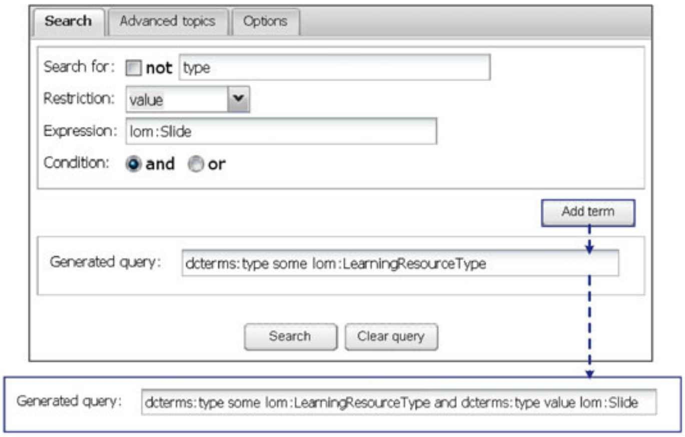
Search for: not type
Restriction: some
Expression: L
Condition:
Generated query: dcterms:type some lom:LearningResourceType

Fig. 6. Formulating a simple query with the guidance of the semantic search interface

The second query refines the previous one, by asking for those items that also satisfy the requirement to be slides. This is expressed as follows:

```
dcterms:type some lom:LearningResourceType and
          dcterms:type value lom:Slide
```

To formulate this query in the semantic search interface, we construct the appropriate class expression representing our new requirement and attach this to the previous one, by checking the 'and' condition (Fig. 7).

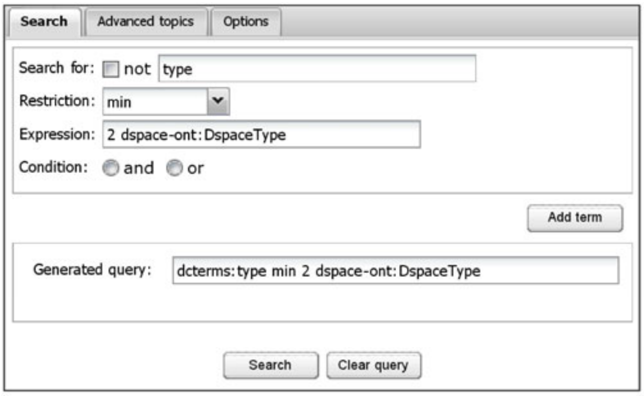


Search for:	[ ] not type
Restriction:	value
Expression:	lom:Slide
Condition:	[✓ and] or
Generated query:	dcterms:type some lom:LearningResourceType
Generated query:	dcterms:type some lom:LearningResourceType and dcterms:type value lom:Slide

Fig. 7. Combining query expressions using the 'and' condition

Finally, we construct a query for retrieving all DSpace items for which we have used more than one DSpace-specific types for their characterization (e.g., learning object and book, presentation and dataset, etc) (Fig. 8).

```
dcterms:type min 2 dspace-ont:DspaceType
```



Search for:	[ ] not type
Restriction:	min
Expression:	2 dspace-ont:DspaceType
Condition:	and or
Generated query:	dcterms:type min 2 dspace-ont:DspaceType

Fig. 8. A more complex query that requires manual typing for the construction of its right atom

In this case the user has to manually input the right query atom where the class name should be accompanied by the required prefix. Note also that such a cardinality-based query cannot be submitted with a traditional, keyword-based mechanism. The same holds for a whole set of other queries that are made possible only through inferencing (see [9] for more examples of such queries).

5 Future Extensions

Currently, the semantic search service is targeted towards guiding novice users in forming simple expressions. For example it is difficult – although not impossible – for someone not familiar with XSD facets to construct composite queries containing numeric or string ranges. The creation of nested queries requires particular attention as well, because the default priority in evaluating a nested expression involving Boolean operators can only be altered using parentheses. For example the expression:

```
dspace-ont:author some dcterms:Agent and dspace-ont:Item
```

evaluates differently than

```
dspace-ont:author some(dcterms:Agent and dspace-ont:Item)
```

Another possible improvement would be to display all class names in the results list. Since query results can belong to more than one class, it would be useful to see all these classes, instead of the most specific one, in the form of a tooltip.

Additionally, more checks and guided options can be added to the user interface, based on what part of the final expression is being defined by the user. For example, the ‘Expression’ field can be controlled depending on what is the user’s choice in the ‘Restriction’ field: ‘some’ and ‘only’ restrictions are always followed by class expressions; ‘value’ needs an individual or literal; and cardinality restrictions (‘min’, ‘max’ and ‘exactly’) are followed by a number and a class expression. Currently, this is circumvented by the ability to give free-text input to the ‘Expression’ box, while auto-complete would work for class names.

In any case, the actual functionality of the system is not hindered, given that the provided query box is editable; therefore someone who is familiar with ontologies and Manchester syntax has no difficulty in proceeding with complex requests.

In addition, more reasoners (e.g. Hermit[7]) and querying approaches, like SPARQL, could be accommodated. Finally, for efficiency and scalability reasons it would be preferable to integrate a persistent semantic storage mechanism with our service. Thus we would be able to support dynamic ontology updates and incremental reasoning, although these techniques are currently well beyond the state of the art.

6 Conclusions

The Semantic Web has grown by the years an extensive technological infrastructure, as it is evident by the increasing number of tools, standards and technologies that build around it. Its success however will be determined by the added-value and

[7] http://hermit-reasoner.com/

tangible gains it brings to the end users. To this extend, not only an adequate number of linked and open information – that would form a "Web of Data" – need to be available, but also efficient and intuitive processes for ingesting this information should be developed. Querying Semantic Web data should not put aside their underlying logic layer either: instead, entailment-based query answering must be integrated and utilized into querying systems, thus bringing the Semantic Web to its full potential.

In this paper we have presented a straightforward approach for querying ontological information by employing the idea of structuring queries through guided input. This necessity comes naturally out of the complexity that is almost inherent in logic-based queries. Besides, current research efforts seem to coincide in trying to alleviate this very problem, no matter what approach do they actually follow – be it text- or formal-based. To our knowledge, this is the first effort to use a DL-based query language that follows the idea of controlled input forms for the structured formulation of semantic queries.

Our prototype has been built as an add-on to the DSpace digital repository system, though by design, the implementation is independent of the system's business logic. In addition, it does not depend on any specific ontology, but can load and interact with any ontology document on the Web. Thus it can serve any ontology-based searching facility or easily integrate with other repository systems or libraries.

Initial user feedback seems promising; however our next step is to make this tool widely available to the community, so as to initiate an extensive evaluation from both the developer and end user perspective.

References

1. Broekstra, J., Kampman, A.: SeRQL: An RDF Query and Transformation Language. In: 3rd International Semantic Web Conference, Japan (2004)
2. Fazzinga, B., Lukasiewicz, T.: Semantic Search on the Web. In: Semantic Web-Interoperability, Usability, Applicability (SWJ), vol. 1, pp. 1–7. IOS Press, Amsterdam (2010)
3. Fikes, R., Hayes, P., Horrocks, I.: OWL-QL - A Language for Deductive Query Answering on the Semantic Web. J. of Web Semantics 2(1), 19–29 (2004)
4. Haarslev, V., Möller, R., Wessel, M.: Querying the Semantic Web with Racer + nRQL. In: International Workshop on Applications of Description Logics (ADL 2004), Germany (2004)
5. Horridge, M., Bechhofer, S.: The OWL API: A Java API for Working with OWL 2 Ontologies. In: 6th OWL Experiences and Directions Workshop, Chantilly, Virginia (2009)
6. Horridge, M., Patel-Schneider, P.S.: Manchester Syntax for OWL 1.1. In: 4th OWL Experiences and Directions Workshop, Gaithersburg, Maryland (2008)
7. Horridge, M., Patel-Schneider, P.S.: OWL 2 Web Ontology Language: Manchester Syntax. W3C Working Group Note (2009), `http://www.w3.org/TR/owl2-manchester-syntax/`
8. Kaufmann, E., Bernstein, A.: How Useful Are Natural Language Interfaces to the Semantic Web for Casual End-Users? 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. 281–294. Springer, Heidelberg (2007)

9. Koutsomitropoulos, D., Solomou, G., Alexopoulos, A., Papatheodorou, T.: Semantic Metadata Interoperability and Inference-Based Querying in Digital Repositories. J. of Information Technology Research 2(2), 37–53 (2009)
10. Koutsomitropoulos, D., Solomou, G., Alexopoulos, A., Papatheodorou, T.: Digital Repositories and the Semantic Web: Semantic Search and Navigation for DSpace. In: 4th International Conference on Open Repositories, Atlanta (2009)
11. Liebig, T., Luther, M., Noppens, O., Wessel, M.: OWLlink. In: Semantic Web-Interoperability, Usability, Applicability, J. IOS Press, Amsterdam (to appear)
12. Lopez, V., Motta, E., Uren, V.: PowerAqua: Fishing the Semantic Web. In: 3rd European Semantic Web Conference, Montenegro (2006)
13. Möller, K., Ambrus, O., Dragan, L., Handschuh, S.: A Visual Interface for Building SPARQL Queries in Konduit. In: 7th International Semantic Web Conference, Germany (2008)
14. Olensky, M.: Semantic interoperability in Europeana. An examination of CIDOC CRM in digital cultural heritage documentation. IEEE Technical Committee on Digital Libraries 6(2) (2010), http://www.ieee-tcdl.org/Bulletin/current/Olensky/olensky.html
15. Prud'hommeaux, E., Seaborne, A. (eds.): SPARQL Query Language for RDF. W3C Recommendation (2008), `http://www.w3.org/TR/rdf-sparql-query/`
16. Seaborne, A.: RDQL - a query language for RDF. W3C member submission (2004), `http://www.w3.org/Submission/RDQL/`
17. Sirin, E., Bulka, B., Smith, M.: Terp: Syntax for OWL-friendly SPARQL Queries. In: 7th OWL Experiences and Directions Workshop, San Francisco (2010)
18. Sirin, E., Parsia, B.: SPARQL-DL: SPARQL Query for OWL-DL. In: 3rd OWL Experiences and Directions, Austria (2007)
19. Smart, P.R., Russell, A., Braines, D., Kalfoglou, Y., Bao, J., Shadbolt, N.R.: A Visual Approach to Semantic Query Design Using a Web-Based Graphical Query Designer. In: Gangemi, A., Euzenat, J. (eds.) EKAW 2008. LNCS (LNAI), vol. 5268, pp. 275–291. Springer, Heidelberg (2008)
20. Zenz, G., Zhou, X., Minack, E., Siberski, W., Nejdl, W.: From Keywords to Semantic Queries - Incremental Query Construction on the Semantic Web. J. Web Semantics 7(3), 166–176 (2009)

Distributed Human Computation Framework for Linked Data Co-reference Resolution

Yang Yang[1], Priyanka Singh[1], Jiadi Yao[1], Ching-man Au Yeung[2], Amir Zareian[1], Xiaowei Wang[1], Zhonglun Cai[1], Manuel Salvadores[1], Nicholas Gibbins[1], Wendy Hall[1], and Nigel Shadbolt[1]

[1] Intelligence, Agents, Multimedia (IAM) Group
School of Electronics and Computer Science
University of Southampton, UK
[2] NTT Communication Science Laboratories
2-4 Hikaridai Seika-cho Soraku-gun
Kyoto, 619-0237, Japan
{yang.yang,ps1w07,jy2e08,ms8,nmg,wh,nrs}@.soton.ac.uk
auyeung@cslab.kecl.ntt.co.jp

Abstract. Distributed Human Computation (DHC) is used to solve computational problems by incorporating the collaborative effort of a large number of humans. It is also a solution to AI-complete problems such as natural language processing. The Semantic Web with its root in AI has many research problems that are considered as AI-complete. E.g. co-reference resolution, which involves determining whether different URIs refer to the same entity, is a significant hurdle to overcome in the realisation of large-scale Semantic Web applications. In this paper, we propose a framework for building a DHC system on top of the Linked Data Cloud to solve various computational problems. To demonstrate the concept, we are focusing on handling the co-reference resolution when integrating distributed datasets. Traditionally machine-learning algorithms are used as a solution for this but they are often computationally expensive, error-prone and do not scale. We designed a DHC system named iamResearcher, which solves the scientific publication author identity co-reference problem when integrating distributed bibliographic datasets. In our system, we aggregated 6 million bibliographic data from various publication repositories. Users can sign up to the system to audit and align their own publications, thus solving the co-reference problem in a distributed manner. The aggregated results are dereferenceable in the Open Linked Data Cloud.

1 Introduction

AI-complete problem is a set of problems found in areas such as image analysis, speech recognition and natural language processing that is difficult for computers to solve effectively but they are relatively easy tasks for humans [14]. Distributed Human Computation (DHC) [11] systems are designed to solve this kind of problems by incorporating collaborative efforts from a large number of

© Springer-Verlag Berlin Heidelberg 2011

humans. This approach is also known as crowdsourcing with computational purpose and in the Web 2.0 term, it's referred as participatory or social systems. For instance, reCAPTCHAs [17] is widely used on the Web to aid transcribing texts of old books that cannot be automatically processed by optical character recognition systems. The Semantic Web is envisioned to be a decentralised worldwide information space for sharing machine-readable data with a minimal cost of integration overheads [13]. However, there are many challenging research problems in the Semantic Web that are considered to be AI-complete, such as co-reference resolution, i.e. determining whether different URIs refer to the same identity [7].

In the recent years, there is an increasing number of linked datasets available on the Web. However, cross-reference and linkage between datasets are sparse as they cannot be easily created automatically. When creating a link between two datasets, intuitively we would consider linking the data that refer to the same thing as a bridge between the two. For instance, DBpedia has a URI referring to one of our authors, Nigel Shadbolt. This can be linked to the URI referring to N. Shadbolt in the Eprints repository dataset because they refer to the same person. Users can then follow the DBpedia URI and find out more about this person's publications. Various machine learning and heuristic algorithms have been proposed to automatically solve this co-referencing problem. However, these approaches are often computationally expensive, error-prone, require some training data, or are difficult to deploy on a large scale.

In this paper, we propose the idea of combining DHC system with Linked Data to create an ecosystem to solve computational problems and facilitate the deployment of Semantic Web. To demonstrate the concept, we focus on the design of a DHC system, iamResearcher[1] that aims to solve the co-referencing problem using DHC.

2 Background

2.1 Co-reference Resolution in the Semantic Web

There are many traditional approaches to perform co-reference resolution on the Web. Besides various natural language processing and machine learning techniques, there are also co-reference resolution systems that are especially designed to use in the Semantic Web to resolve URIs and name ambiguities.

In the area of machine learning Soon et al [16] resolved noun phrases by creating a co-reference relation and measuring the distance between two identities in order to find matches between nouns. Ng et al. [10] improved their algorithm by including more sophisticated linguistic knowledge to improve precision. In both cases, the authors found that performance dropped significantly when the dataset became larger and human intervention was required to solve co-references that were not accurately resolved automatically. Regarding author name ambiguities,

[1] http://www.iamresearcher.soton.ac.uk/ for University of Southampton members access only, and http://www.iamreseacrher.com for Global users.

Kang et al [8] had shown that co-authorship is a very reliable and decisive method to validate the identity of an author when there were namesakes. They proposed that author name disambiguation can be solved by clustering similar names into groups of identities and making use of other available information such as email addresses and publication titles to resolve the issue.

While, in an ideal Semantic Web, the identity of one person may be represented by different URIs in different systems. Sleeman et al. [15] proposed to use a rules-based model and a vector space model to cluster entities into groups of co-references. Whereas, Glaser et al [4] proposed the Co-reference Resolution Service to facilitate management of URI co-references. Salvadores et al [12] used LinksB2N algorithm to discover overlapping RDF data repositories to integrate datasets using clustering technique to find equivalent data.

These methods somewhat solves the co-reference problem but Semantic Web contains many highly complex data and these algorithms are insufficient in addressing the distinction between two URIs when they represent different entities in different context.

2.2 Human Computation

Human computation is a method of making use of the collaborative effort of a large number of humans to solve problems that are still difficult for computers to perform accurately. These tasks include natural language processing, speech recognition and image processing, which are relatively easy for human beings. Nowadays, people are more engaged into social activities on the Web, they collaborate and share information with one another, Wikipedia and Twitter are some of the many examples. The combination of the Social Web and human computation provides many opportunities to solve difficult computational problems.

reCAPTCHA, a system for distinguishing between humans and automated computer bots on the Web and at the same time it helps the digitization of millions of books [17] is a popular example of DHC. It proves that when a proper monitoring process is available and when users have the motivation or incentive to use such a system, one can collect reliable information for solving difficult computational problems. Albors et al. [1] discussed about the evolution of information through the effort of crowdsourcing. They mentioned Wikipedia and folksonomies as examples, where users are both the creators and consumers of the shared data, thus creating an ecosystem for the growth and development of information that ultimately benefit the users themselves. Gruber [6] discussed the structure of a collective knowledge system in which users interacted with one another and contributed their knowledge, while machines facilitated communications and retrieval of resources in the background, aggregating the contributions of individual users.

The following section discusses the implementation of DHC framework to solve the co-reference resolution in our system.

3 Linked Data Ecosystem Framework

At present there are 203 RDF datasets that have been published on the Linked Data Cloud.[2] Although this is encouraging, we are still relatively far from the Semantic Web envisioned by Tim Berners-Lee [2]. There are still many challenging research questions that are needed to be solved. From our experience in carrying out the Enakting project,[3] whose goal is to build the pragmatic Semantic Web, we have identified several challenges in Linked Data, such as co-reference resolution, ontology mapping, resource discovery, and federated query from multiple datasets [9]. Many research efforts in the past have been devoted to develop heuristic machine learning algorithm to solve these problems. However, these automated solutions do not necessarily solve these problems accurately.

Here, we propose to solve these problems by using DHC approach to build linked data ecosystem in which difficult computational tasks are distributed to the users in the system. And by ecosystem we mean that it is a self-sufficient system that can provide a long-term solution to a particular problem. For instance, an automated Semantic Web reasoner is likely to fail to return an answer when querying incomplete or noisy data. One can imagine a DHC system that can overcome this problem by enabling distributed reasoning on a subset of data with facilitation from human in certain decision making processes.

By studying different DHC systems, we have identified a list of common characteristics and designed a Linked Data Ecosystem Framework as depicted in Figure 1. To design an ecosystem first we need to identify the system stakeholders, i.e. the target data consumers and publishers. Next, we have the four major components for sustainability, namely incentive, human interface, data aggregation and quality control.

Incentive. We need to make sure that users have the incentive to use the system and therefore contribute to solving the problem which can manifest in different modality in different system. For instance, users want to use a system because they get paid, gain reputation, or simply because it is fun to use. This requires anthropological studies of the system stakeholders and we can design the system based on analysis of the generic usefulness of the system for the targeted crowd.

Human Interface to solve computational problem. This is the core of the system. It requires an interface that is applicable to the individual or small group of people to solve a computational problem in a distributed manner. For many problems, the system can use heuristic method to automate certain tasks to assist the human contributors.

Aggregation. The system combines the distributed human computation and heuristic algorithm output and aggregate the results to solve the global problem.

[2] http://www4.wiwiss.fu-berlin.de/lodcloud/state/ as on 22nd September 2010
[3] http://www.enakting.org

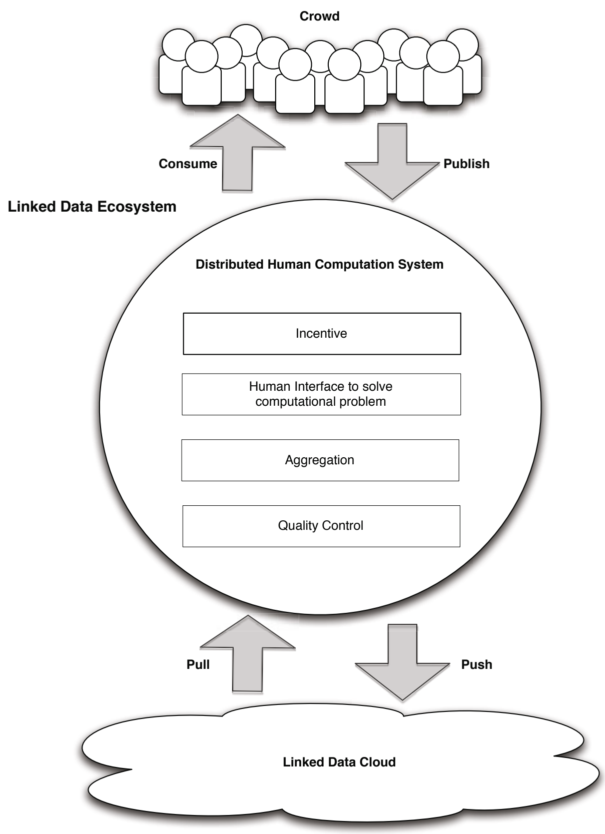


Fig. 1. Linked Data Ecosystem Framework

Quality Control. How does the system cope with possibility of fraud or incorrect answers to ensure some level of quality in the solution to the overall problems? The quality control in this framework acts as a layer to ensure the quality of the data to be pushed into the Linked Data Cloud.

In the following section, we apply the framework to a specific scenario–solving the co-reference problem in linked data.

4 Designing iamResearcher

The co-reference problem we are trying to solve in this paper is the name ambiguity problem in distributed digital libraries. Here is a typical scenario. In the Eprints[4] repository we have a list of publications authored by a person named Wendy Hall. In the PubMed[5] repository we have another list of publications

[4] http://www.eprints.org/

[5] http://www.ncbi.nlm.nih.gov/pubmed

authored by a person named W. Hall. If we want to design an expert finder algorithm that can rank researchers' expertise based on their publications, we must decide whether these two names refer to the same person.

Most of the large scale digital library repositories nowadays are not capable of resolving co-referencing and ambiguities. This is because it is difficult to determine if W. Hall is Wendy Hall or William Hall. Names of researchers are usually incomplete or inconsistent across different digital libraries. In particular, the name can be written with different combinations of the initials, the first name, middle name and last name. There can even be incorrect spellings. For example, within our own institutional Eprints repository, there can be as many as six different ways of naming any individual author. The extent of this name ambiguity can be seen within the UK research community based on the analysis of the Research Assessment Exercise 2001 records we did in the previous AKT project.[6] Within the list of researcher names in the institutional submissions, 10% of names lead to clashes between two or more individuals. If the names are restricted to a single initial, the proportion of clashes rises to 17% [5]. This situation can be more severe on the global scale. The VIAF project[7] also designed a service to integrate different global libraries using a heuristic name-matching algorithm in bibliographic record clustering allowing national and regional variation which is difficult to make an alignment.

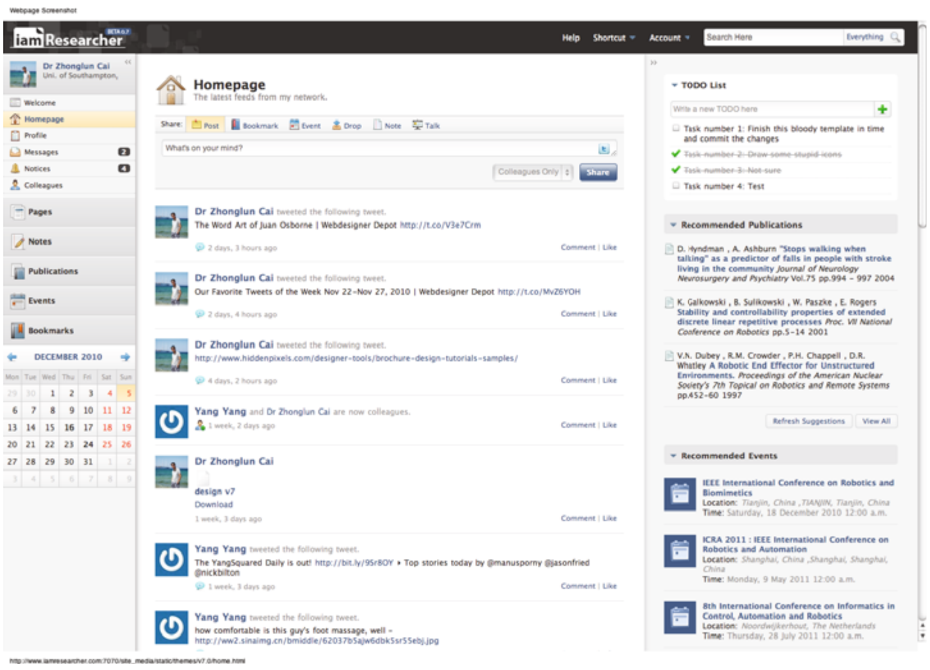
Co-reference problem has been well studied in computational linguistics. How do we determine if two things are the same? Leibniz's Law [3] states that 'X is the same as Y if, and only if X and Y have all the same properties and relations; thus, whatever is true of X is also true of Y, and vice-versa'. Based on this concept, we can compare the identities' relations and properties to determine if they are the same. For instance, we can check whether two names have the same affiliation and the same email address. However, in the real world, different information can be missing in different publication repositories. Even when all the information is available for comparison, one still have to consider the fact that properties of the same person can change over time. For example, when a researcher moves from one institution to another, his/her email address is likely to change.

As mentioned before, in order to derive the correct interpretation of a name, it should be connected to the right individuals. Therefore we propose to link the publication data with its individual author to solve the name ambiguity problem as the author would have the best knowledge about their own publication. Therefore, the fundamental ideas is that we aggregate bibliographic data from various repositories and ask users to audit the data and make alignment with their own publication data. This solves the name ambiguity co-reference problem.

Applying our framework, first we need to identify the stakeholders in our system–the data publishers and consumers. In our case, researchers play both roles. With above requirements, we designed a system that link all researchers and publications together. The system automatically pulls out all publications

[6] http://www.aktors.org/akt/

[7] http://www.viaf.org



Mon	Tue	Wed	Thu	Fri	Sat	Sun
29	30	1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31	1	2
3	4	5	6	7	8	9

Dr Zhonglun Cai tweeted the following tweet.
The Word Art of Juan Osborne | Webdesigner Depot http://t.co/V3e7Crm
2 days, 3 hours ago

Dr Zhonglun Cai tweeted the following tweet.
Our Favorite Tweets of the Week Nov 22–Nov 27, 2010 | Webdesigner Depot http://t.co/MvZ6YOH
2 days, 4 hours ago

Dr Zhonglun Cai tweeted the following tweet.
http://www.hiddenpixels.com/designer-tools/brochure-design-tutorials-samples/
4 days, 2 hours ago

Yang Yang and Dr Zhonglun Cai are now colleagues.
1 week, 2 days ago

Dr Zhonglun Cai
design v7
Download
1 week, 3 days ago

Yang Yang tweeted the following tweet.
The YangSquared Daily is out! http://bit.ly/9SrBOY ▸ Top stories today by @manusporny @jasonfried @nickbilton
1 week, 3 days ago

Yang Yang tweeted the following tweet.
how comfortable is this guy's foot massage, well – http://ww2.sinaimg.cn/bmiddle/62037b5ajw6dbk5sr55ebj.jpg

- ☐ Task number 1: Finish this bloody template in time and commit the changes
- ✓ ~~Task number 2: Draw some stupid icons~~
- ✓ ~~Task number 3: Not sure~~
- ☐ Task number 4: Test

D. Hyndman , A. Ashburn "Stops walking when talking" as a predictor of falls in people with stroke living in the community *Journal of Neurology Neurosurgery and Psychiatry* Vol.75 pp.994 – 997 2004

K. Galkowski , B. Sulikowski , W. Paszke , E. Rogers Stability and controllability properties of extended discrete linear repetitive processes *Proc. VII National Conference on Robotics* pp.5–14 2001

V.N. Dubey , R.M. Crowder , P.H. Chappell , D.R. Whatley A Robotic End Effector for Unstructured Environments. *Proceedings of the American Nuclear Society's 7th Topical on Robotics and Remote Systems* pp.452–60 1997

IEEE International Conference on Robotics and Biomimetics
Location: *Tianjin, China ,TIANJIN, Tianjin, China*
Time: Saturday, 18 December 2010 12:00 a.m.

ICRA 2011 : IEEE International Conference on Robotics and Automation
Location: *Shanghai, China ,Shanghai, Shanghai, China*
Time: Monday, 9 May 2011 12:00 a.m.

8th International Conference on Informatics in Control, Automation and Robotics
Location: *Noordwijkerhout, The Netherlands*
Time: Thursday, 28 July 2011 12:00 a.m.

Fig. 2. iamResearcher User Homepage

from various resources (as mentioned in Table 1) for researchers to audit and align the data. By analysing the network graph, researchers are then linked to each other via the co-authorships of the publications. The co-authorship often reflects their professional social network - if you often write papers with certain set of authors, most likely they are your colleagues. Based on this we designed a professional network portal-like application [18] - Researchers signup on the system to find their publications and establish the colleagueship with their co-authors and so on.

The general incentive for data consumer to use the system is that they can find experts and publications in their research field. The general incentive for the data publishers to use the system is that by creating their own list of publications they enable other researcher to find, read and cite their work. A researcher's scientific publication can evidently reflect his/her expertise. Therefore, individual may also be motivated to set up a list of publication for this purpose as well. To amplify the usage of the system, we also designed list of generic researcher oriented services like publication and research events recommendation based on their research interests, easy communication with their colleagues, group management system, bookmark management system etc. to encourage researcher to collaborate and use the system on the daily basis. We also make the user's

FOAF profile with their publications dereferenceable in the Open Linked Data Cloud. Figure 2 illustrates the homepage of the system showing status updates from their colleagues and recommended publications and conferences.

In the following section, we will elaborate the system design of the co-reference management and how we deal with quality control issues.

5 Co-reference Resolution

We have harvested metadata of publications from various repositories and databases. Table 1 gives an overview of the data we have collected.

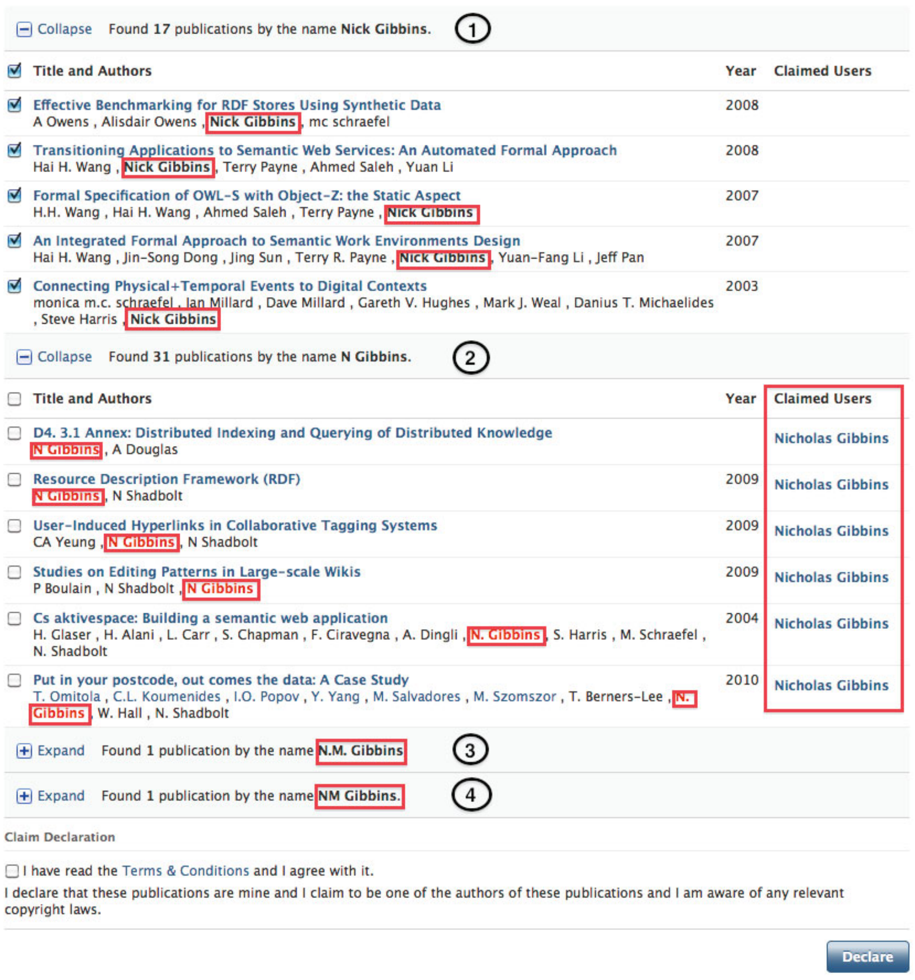
Table 1. Dataset Source

Source	**Subjects Cover**	**Paper's Source**	**Papers Extracted**
PubMed	Life sciences, Medicine	Peer-reviewed journal articles	1381081
Institutional EPrints	Multi-discipline	Preprint papers uploaded by researchers from each institute	203387
arXiv	Mathematics, Physics and Biology	Preprint papers uploaded by researchers	478092
DBLP	Computer science	Papers harvested from VLDB, IEEE, ACM	1394314
Econpapers	Economics	Part of RePEc	361224
Citeseer	Information Sciences, Engineering Sciences	Papers harvest from the web according to rules	345821
PANGAEA	Geoscientific and Environmental Sciences	Data submitted by researchers across the world	576939
Others	Multi-discipline	Papers harvested from search engine and numerous databases	213276

Our co-reference system is designed as a two-stage process. Firstly, we used heuristic name matching algorithms to pull out all the possible combination and spelling of authors' names. Secondly, we let users audit the data by allowing them to select the publications from the resulted list.

When users register an account, we ask for their first and last name, our interface clearly states not to enter fake names or aliases as the system use their names to search for their publications and an incorrect name would lead the system in getting no matches or wrong matches.

The name-matching algorithm performs three types of matches: full name match, exact initial match and loose initial match.



⊟ Collapse Found **17** publications by the name **Nick Gibbins.** 1

☑ Title and Authors	Year	Claimed Users
☑ Effective Benchmarking for RDF Stores Using Synthetic Data A Owens , Alisdair Owens , Nick Gibbins , mc schraefel	2008	
☑ Transitioning Applications to Semantic Web Services: An Automated Formal Approach Hai H. Wang , Nick Gibbins , Terry Payne , Ahmed Saleh , Yuan Li	2008	
☑ Formal Specification of OWL-S with Object-Z: the Static Aspect H.H. Wang , Hai H. Wang , Ahmed Saleh , Terry Payne , Nick Gibbins	2007	
☑ An Integrated Formal Approach to Semantic Work Environments Design Hai H. Wang , Jin-Song Dong , Jing Sun , Terry R. Payne , Nick Gibbins , Yuan-Fang Li , Jeff Pan	2007	
☑ Connecting Physical+Temporal Events to Digital Contexts monica m.c. schraefel , Ian Millard , Dave Millard , Gareth V. Hughes , Mark J. Weal , Danius T. Michaelides , Steve Harris , Nick Gibbins	2003	

⊟ Collapse Found **31** publications by the name **N Gibbins.** 2

☐ Title and Authors	Year	Claimed Users
☐ D4. 3.1 Annex: Distributed Indexing and Querying of Distributed Knowledge N Gibbins , A Douglas		Nicholas Gibbins
☐ Resource Description Framework (RDF) N Gibbins , N Shadbolt	2009	Nicholas Gibbins
☐ User-Induced Hyperlinks in Collaborative Tagging Systems CA Yeung , N Gibbins , N Shadbolt	2009	Nicholas Gibbins
☐ Studies on Editing Patterns in Large-scale Wikis P Boulain , N Shadbolt , N Gibbins	2009	Nicholas Gibbins
☐ Cs aktivespace: Building a semantic web application H. Glaser , H. Alani , L. Carr , S. Chapman , F. Ciravegna , A. Dingli , N. Gibbins , S. Harris , M. Schraefel , N. Shadbolt	2004	Nicholas Gibbins
☐ Put in your postcode, out comes the data: A Case Study T. Omitola , C.L. Koumenides , I.O. Popov , Y. Yang , M. Salvadores , M. Szomszor , T. Berners-Lee , N. Gibbins , W. Hall , N. Shadbolt	2010	Nicholas Gibbins

⊞ Expand Found 1 publication by the name N.M. Gibbins 3

⊞ Expand Found 1 publication by the name NM Gibbins. 4

Claim Declaration

☐ I have read the Terms & Conditions and I agree with it.

I declare that these publications are mine and I claim to be one of the authors of these publications and I am aware of any relevant copyright laws.

Fig. 3. User Auditing Interface. Caption (1)- Full Name Matching. Caption (2)- Exact Initial matching. Caption (3)(4)- Loose Initial Matching

Full name matching. This makes two matches:

* It finds papers with an exact match of user's name with publication's author's name.
* It matches when author's first name starts with user's first name.

For example, author Nick Gibbins, Nick A. Gibbins can be matched with user profile name Nick Gibbins. We group these result together and pre-select them as it shown in Figure 3 point (1).

Exact initial matching. Many authors' names in our dataset are not in full, instead, they are written in initial with their last name format. This finds papers that matches user's initial and last name. The initial of the user is computed by taking the first letter of the first name. We put these results in one group as it is illustrated in Figure 3 point (2) and it is not pre-selected because in most cases the results are from multiple authors. The figure also demonstrates a special case, where there is a user named Nicholas Gibbins who had already claimed some of the publications, the system highlights them to make distinction from the publications that are free to claim and the publications that have already been claimed by other users. If there is a wrongful claim or the claimed author is an impostor, user can follow the link to view the claimed author's profile details and can even report fraud.

Loose initial matching. We take the initial and last name of all the authors in our database and match it with the current user's. This match finds authors that have multi-letter initials. As it is shown in Figure 3 point (3) and (4) there are two more matches - N.M Gibbins, NM Gibbins. We collapse this group of results for a cleaner interface, as there can be multiple results.

Some of our publication records also have email address associated with them, which can be a very accurate property to find user's publications. Therefore, we also enable users to enter all email addresses they use to publish their papers to do an automated pre selection of the paper as an option. For some special case, for instance when user has a different name associated with different publication, they can search the single publication and make a claim. This also holds true for misspelling or any other foreseen errors in the publications, user can simply search for them separately or add and even edit the publication themselves by the service 'Add or Edit Publication' provided by the system. When our publication database is updated or someone enters a new publication, users are notified to update their publication list as well.

In our system publications are modelled by using the Bibliographic Ontology[8] and the author of the publication is modelled by Dublin Core metadata[9]. The URI `http://www.iamresearcher.com/publication/rdf /1661006/` illustrates a single publication record. When user signup on our system, we generate a unique URI for each user and model their profile and their social relations by using FOAF ontology. When user claims a publication, they make alignment of the publication and their FOAF URI. These data is then pushed into the Linked Data Cloud and is dereferencable, e.g. by dereferencing the URI of this user `http://www.iamresearcher.com/profiles/id/yang2/` you will get an RDF file with list of the publications this user has claimed to be the author of. Following the publication links provided in the RDF an agent can easily pull out a user's co-author network graph and so on.

Our system is designed in a way that anyone can claim to be an author of any publication. Users are asked to agree to our terms and conditions as the

[8] `http://purl.org/ontology/bilbo/`
[9] `http://purl.org/dc/`

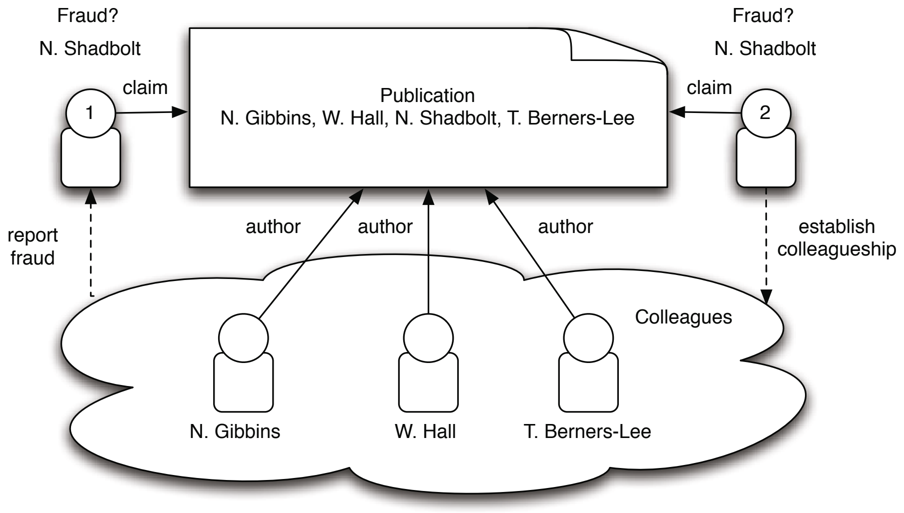


Fig. 4. Quality Cotnrol: Report of Fraud

system does not take any responsibility of breach of copyright or intellectual property issues, users who claim the publications are responsible for all the legal matters. So we enable users to report spam and fraud to maintain system integrity. The social network application has the benefit to identify a real user from fake by analysing their network structure and by examining the FOAF ontology. When dereferencing a user's FOAF URI, we get RDF to describe this identity, besides some basic information, we are defined by whom we related to. In our system, if a publication has five authors, it will be audited five times by all the authors. As we mentioned before, if you have co-authored a publication with someone, there is strong possibility that they are your colleagues too and you may want to establish the professional and social relationship as well. So in most of the scenario, we can easily identify a fraud because an impostor would fail to establish social relationships with other researchers.

Figure 4 illustrates how our system can spot a fraud. In this diagram, N. Gibbins, W. Hall and T. Berners-Lee claimed this particular publication, they have also established colleagueship in the FOAF file. Assume there are two users whose names are N. Shadbolt and they both claim to be author of same publication. How do we identify who is a fraud? As soon as one of them establishes a social relationship with any of the existing claimed author, others can spot and report the fraud. Indeed, someone can pretend to be someone to add social relationship as well, but it would be eventually spotted by observing day-to-day communication through the social network. For this purpose and for the convenience of users, we have designed a co-author invitation claim. So users can invite their co-authors to claim the publication and keep the integrity of the publication and in result the whole system.

6 Preliminary Evaluation Results

We deployed our system at the University of Southampton for evaluation. In this trial, we mainly focused on measuring two factors of the system: the incentive and total number of publications users claimed in comparison to the publication they deposited in the University Eprints repository. We sent emails to three different research labs at the University. We advertised the system as a free research platform and provided a link to the system. 163 users signed up initially (many of them were research students), we chose the 52 users who had deposited publication in the University repository as case study. As it is shown in Table 2, 39 out of these 52 users had claimed publications. This shows our system successfully used incentive as 75% users claimed their publications. Few research students gave feedback that they were pleased with the personalised recommendation system and easily found their publication. As our system aggregated data from the University repository (Eprints) and other repositories as well ([1]), our dataset is a superset of the University repository. By comparing the claimed publications, we estimated how well a user solved the co-reference problem in their own publications. In our analysis we found that, out of 39 users who had claimed publications, 51% of them claimed 136% more publications than they deposited at Eprints. It proves the success of the system as users claimed 39% more publications aggregated from different sources grouped together than the one where they entered bibliographical data in the Eprints repository themselves and also solved the co-reference problem in the integrated dataset. In contrast, 49% of users did not claim all of their publications they deposited in Eprints and only 67% of their publications were claimed. In the group of these users, we observed that many of them had a large amount of publications, for instance, one of the user had 417 publications, where he/she only claimed 289 pre-selected results. Claiming publications can be time consuming so our system also provides options for users to claim publication not only during registration process but also later at their own convenient time. Due to the time limitation, we did not observe the users for longer period to identify how many of them claimed publications later when system notified them to update their publication list. However, we believe if we deploy the system to a larger demographic, our system would produce even more promising results fuelled by the network effect.

Table 2. User Claim Rate. Total Users(A): the amount of users registered to use our system within our University; Users have published(B): Amount of users who had publications in the University repository; Users claimed(C): From users in B, the amount of users who had claimed publications in our system; Percentage: C/B, percentage of users who had committed work to our system.

Total Users(A)	**Users have published(B)**	**Users claimed (C)**	**Percentage (D)**
163	52	39	75%

Table 3. Decomposition of users who claimed publications. Claimed Pubs(B): Claimed publications by that group of users; Univ. Repos(C): Amount of publications found in University repository for that group of users; Claim Perc(B/C): B/C, Claim percentage of that group of users. Perc of Users: Percentage of users who had made a claim. Since our dataset is a superset of the University repository, these users are presented with at least those publications that can be found in the University repository. If a user claims all the publication he deposited in the university repository, his Claim Perc would be 100%. This table splits those who under-claimed from the claimed authors. Therefore, those 19 users (bottom row), who on average claimed only 67% of the repository total, did not put in enough effort to find their publications in our system, while as the other 20 users (top row), who on average claimed 136% more that the university repository, managed to find publications we aggregated from other databases.

No. of Users	**Claimed Pubs(B)**	**Univ. Repos(C)**	**Claim Perc(B/C)**	**Perc of Users**
20	1349	991	136%	51%
19	613	921	67%	49%

7 Conclusions

In some cases machines are not capable of solving the problems that are easier for humans and in our system we have taken the best of both worlds, the computational power of machines and cognitive ability of humans and brought them together to create a distributed human computation system to solve the Linked Data issue of co-referencing. This system creates an ecosystem by making the users, in this case researchers, the creators and consumers of data. Moreover, the platform we provide allows them to make a complete cycle of resource utilisation and consumption.

We have also emphasised the importance of incentive to motivate the user to contribute to the system and made a trustworthy structure to identify fraud and stop spam. But it is necessary to mention that since the system heavily relies on the human interaction and contribution, any shortcomings of humans is the shortcoming of this system as well. For example, if users fail to contribute then the system is unable to fix the errors, the system is as smart as the users using it and as diverse as the community of people. Also, as we take advantage of the researcher's social network, there is a risk of incomplete auditing of data when there are very few users from a network or research fields and we also need to take account of the old researchers who are no longer working and part of the research community or who are no longer active.

Finally, since the system is an ecosystem there must be equilibrium, the numbers of users and data are directly proportional, if there is scarcity of users or the data, the system will fail. And as it happens with any other natural ecosystem, it is vulnerable to unforeseen external factors and loose the balance to function properly as a stand-alone system. But with strong community contribution and support, the system can be resilient and thrive to become a stable and dynamic environment to provide better computation.

Acknowledgements

This work was supported by the EnAKTing project under grant EP/G0088493/1 and Web Science Doctoral Training Centre under grant EP/G036926/1, funded by the Engineering and Physical Sciences Research Council and the Research Innovation Services at the University of Southampton.

References

1. Albors, J., Ramos, J.C., Hervas, J.L.: New learning network paradigms: Communities of objectives, crowdsourcing, wikis and open source. International Journal of Information Management 28(3), 194–202 (2008)
2. Berners-Lee, T., Hendler, J., Lassila, O.: The semantic web: Scientific american. Scientific American 284(5), 34–43 (2001)
3. Feldman, F.: Leibniz and" Leibniz'Law". The Philosophical Review 79(4), 510–522 (1970)
4. Glaser, H., Jaffri, A., Millard, I.: Managing co-reference on the semantic web. In: WWW 2009 Workshop: Linked Data on the Web (LDOW 2009) (April 2009)
5. Glaser, H., Lewy, T., Millard, I., Dowling, B.: On coreference and the semantic web (December 2007)
6. Gruber, T.: Collective knowledge systems: Where the social web meets the semantic web. Web Semantics: Science, Services and Agents on the World Wide Web 6(1), 4–13 (2008), Semantic Web and Web 2.0
7. Jaffri, A., Glaser, H., Millard, I.: Uri identity management for semantic web data integration and linkage. In: On the Move to Meaningful Internet Systems 2007: OTM 2007 Workshops. LNCS, pp. 1125–1134. Springer, Heidelberg (2007)
8. Kang, I.-S., Na, S.-H., Lee, S., Jung, H., Kim, P., Sung, W.-K., Lee, J.-H.: On co-authorship for author disambiguation. Information Processing and Management 45(1), 84–97 (2009)
9. Millard, I., Glaser, H., Salvadores, M., Shadbolt, N.: Consuming multiple linked data sources: Challenges and Experiences (November 2010)
10. Ng, V., Cardie, C.: Improving machine learning approaches to coreference resolution. In: Proceedings of the 40th Annual Meeting on Association for Computational Linguistics, ACL 2002, pp. 104–111. Association for Computational Linguistics, Morristown (2002)
11. Quinn, A.J., Bederson, B.B.: A taxonomy of distributed human computation. Human-Computer Interaction Lab. Tech Report, University of Maryland (2009)
12. Salvadores, M., Correndo, G., Rodriguez-Castro, B., Gibbins, N., Darlington, J., Shadbolt, N.R.: LinksB2N: Automatic data integration for the semantic web. In: Meersman, R., Dillon, T., Herrero, P. (eds.) OTM 2009. LNCS, vol. 5871, pp. 1121–1138. Springer, Heidelberg (2009)
13. Shadbolt, N., Hall, W., Berners-Lee, T.: The semantic web revisited. Intelligent Systems 21(3), 96–101 (2006)

14. Shapiro, S.C.: Encyclopedia of artificial intelligence, vol. 1,2 (1992)
15. 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 (November 2010)
16. Soon, W.M., Ng, H.T., Lim, D.C.Y.: A machine learning approach to coreference resolution of noun phrases. Computational Linguistics 27(4), 521–544 (2001)
17. von Ahn, L., Blum, M., Hopper, N., Langford, J.: Captcha: Using hard ai problems for security. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 646–646. Springer, Heidelberg (2003)
18. Yang, Y., Yeung, C.M.A., Weal, M.J., Davis, H.: The researcher social network: A social network based on metadata of scientific publications. In: Proceedings of WebSci 2009: Society On-Line (March 2009)

Relational Kernel Machines for Learning from Graph-Structured RDF Data*

Veli Bicer[1], Thanh Tran[2], and Anna Gossen[3]

[1] FZI Forschungszentrum Informatik, Haid-und-Neu-Str. 10-14,
76131 Karlsruhe, Germany
bicer@fzi.de
[2] Institute AIFB, Geb. 11.40 KIT-Campus Sd, 76128 Karlsruhe, Germany
duc.tran@kit.edu
[3] Fluid Operations AG, Altrottstr. 31, 69190 Walldorf, Germany
anna.gossen@fluidops.com

Abstract. Despite the increased awareness that exploiting the large amount of semantic data requires statistics-based inference capabilities, only little work can be found on this direction in the Semantic Web research. On semantic data, supervised approaches, particularly kernel-based Support Vector Machines (SVM), are promising. However, obtaining the right features to be used in kernels is an open problem because the amount of features that can be extracted from the complex structure of semantic data might be very large. Further, combining several kernels can help to deal with efficiency and data sparsity but creates the additional challenge of identifying and joining different subsets of features or kernels, respectively. In this work, we solve these two problems by employing the strategy of *dynamic feature construction* to compute a hypothesis, representing the relevant features for a set of examples. Then, a composite kernel is obtained from a set of *clause kernels* derived from components of the hypothesis. The learning of the hypothesis and kernel(s) is performed in an interleaving fashion. Based on experiments on real-world datasets, we show that the resulting *relational kernel machine* improves the SVM baseline.

1 Introduction

The amount of semantic data captured in ontologies or made publicly available as RDF Linked Data is large and ever increasing. While deductive reasoning has been the research focus so far, and is desirable for some cases, it has become common belief that the scale of semantic data also demands for more efficient statistics-based techniques.

Despites the increased awareness that dealing with the large amount of semantic data benefits from statistics-based inference capabilities, only the few

* This research was funded by means of the German Federal Ministry of Economy and Technology under the promotional reference 01MQ07012. The authors take the responsibility for the contents.

© Springer-Verlag Berlin Heidelberg 2011

seminal approaches mentioned above can be found. While they show how existing techniques can be applied semantic data, many challenges specific to this scenario remain unresolved.

That dealing with the large amount of semantic data benefits from statistics-based inference capabilities, only little work can be found in Semantic Web research. Basically, semantic data can be conceived as a graph where nodes stand for resources and edges capture attribute values or relations between resources. The main machine learning (ML) approaches in the field of *relational learning* that directly operate on relational and graph-structured data of this kind are *Inductive Logic Programming* (ILP) and *Statistical Relational Learning* (SRL). In [7] the authors propose the use of existing SRL approaches and their integration into SPARQL such that both deductive and inductive reasoning can be employed for query answering. Also, it has been shown how ontologies can be exploited as prior knowledge to improve SRL [11]. Lastly, most relevant for this paper is the work on using kernels and *Support Vector Machines* (SVM) for learning on semantic data [4,3,1].

While unsupervised SRL approaches provide a probability distribution over the entire set of input data (e.g. the entire RDF graph) for a possibly large amount of random variables, supervised approaches such as ILP and SVM are conducted to make a prediction for one single random variable. While the former is thus more flexible and can be used to infer different kinds of knowledge, the latter focuses on one single prediction problem and as a result, exhibits better scalability and mostly, also better predictive performance. In fact, kernel-based SVM is an important topic in machine learning (ML) due to the robustness of SVM and the fact that kernels flexibly allows for data of arbitrary structure (e.g. as complex as graphs) to be exploited for learning. In particular, it is promising for learning on semantic data [4,3,1]. Following this line of research, we address two major problems:

Problem 1. Basically, a kernel measures the similarity of two data points based on their features. *Obtaining the right features* is a classic ML problem, which is exacerbated in the semantic data setting. Here, a data point (e.g. a RDF resource or a triple) might have a large set of features (e.g. all attributes of a resource, all related resources and all those related to them). In the extreme case, the features for every RDF resource correspond to the entire RDF graph. Basically, a preprocessing technique can materialize all the possible features derived from relations between the resources and store them in a single table. However, incorporating all possible features like this is expensive and that might be not affordable when the data is too large. While the previous work on using SVM on semantic data also incorporates complex kernels that can take relations between resources into account, it does not answer the question which relations are to be used for implementing such kernels [4,3,1]. In fact, not only SVM but also the mentioned SRL approach [7] require features to be chosen manually by an expert. More precisely, the latter does not train the model on the entire dataset but focuses on the 'features' defined in the SPARQL query.

Problem 2. Related to this is the problem of *combining different sets of features*. The authors in [1] already made clear that several kernels are needed for considering different kinds of similarities, and proposed a simple weighted additive combination. Generally, instead of one single kernel, multiple ones might be used for reasons of scalability and data sparsity. The goal is to identify and apply kernels only on some non-sparse subsets of the data. However, not only the question which kernels are to be used but also the tuning of their weights are left open in [1].

Contributions. We address these two problems of applying kernel-based SVM on semantic and relational data. The contributions can be summarized as follows:

– We present a novel approach that combines relational learning and kernel-based SVM learning to obtain what we call *Relational Kernel Machines* for graph-structured data. For this, we consider learning as a classification problem in which the aim is to train a kernel machine from example data points whose feature sets have to be extracted from the data graph.
– In particular, we employ the strategy of *dynamic feature constructions* used in ILP to compute relevant features by iteratively searching for a hypothesis that covers a given set of examples. The hypothesis is a set of clauses, each can be conceived as an intensional description of a subset of relevant features of the examples.
– We propose a R-convolution kernel called the *clause kernel*, which is created for every clause of the hypothesis and combined to a form of a *composite kernel*, the basis of our kernel machine.
– The technical problem behind this is that the search for a valid hypothesis (i.e., finding relevant features) and the training of the kernel machine (i.e., finding the combination of features/kernels and incorporating it into the SVM) are two dependent processes that need to be solved concurrently. We propose a *coevolution-based genetic algorithm* to solve this multi-objective optimization.

Structure. We introduce the basic concepts in Section 2. Then, our approach is presented in Section 3, where we start with the clause kernels, then discuss how they can be combined and optimized using coevolution, and finally show how the resulting kernel machines can be applied. The experiments are presented in Section 4, followed by related work in Section 5 and conclusion in Section 6.

2 Preliminaries

Learning with Kernel Machines. In this work we consider the task of classification as in the case of a Support Vector Machine (SVM). Assume a dataset $\mathcal{D} = \{(x_1, y_1), .., (x_n, y_n)\}$ with data point samples $x_i \in \mathcal{X}$ and $y_i \in \{+1, -1\}$. Assume that data points x_i are part of a graph-structured data model $G = (V, E, l)$, where $V = \{v_1, ..., v_n\}$ is a finite set of vertices, $E \subseteq V \times \mathcal{L}_E \times V$ is a set of edges, and $l : V \to \mathcal{L}_V$ is a vertex labeling function, given a set of vertex labels $\mathcal{L}_V$ and a set of edge labels $\mathcal{L}_E$. This model might capture semantic data in the form of RDF or relational data. In this paper, we focus on graph-structured RDF data.

In this context, a data point might be a RDF resource, or a RDF triple and the task here is to predict whether a given resource belongs to a class or a pair of resources instantiate a triple pattern $p(\cdot,\cdot)$ (henceforth called target predicate). In particular, we consider a data point x_i as a node pair $x_i = \langle v_1^i, v_2^i \rangle$ and decompose the dataset into positive and negative examples, such that for a target predicate $p(\cdot,\cdot)$, we have $\mathcal{D}^+ = \{\langle v_1^i, v_2^i \rangle \mid p(v_1^i, v_2^i) \in E\}$ and $\mathcal{D}^- = \mathcal{D} \backslash \mathcal{D}^+$. Thus, every data point x_i is represented as a pair of vertices, and its features are captured by edges to other vertices in the graph. Clearly, such a data point models an RDF triple, and also this notion carries over to the case of RDF resources, e.g. every resource v can be conceived as a triple $Thing(v, type)$ and modeled as $\langle v, Thing \rangle$.

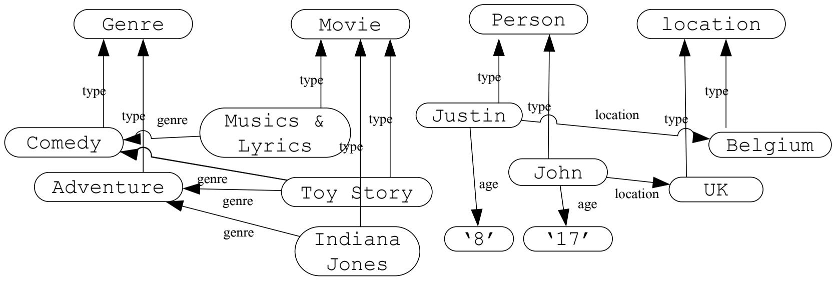


Fig. 1. An example data graph

An example data graph is illustrated in Fig. 1. A dataset over this graph for the target predicate $likes(\cdot,\cdot)$ is $\mathcal{D} = \{(\langle Justin, ToyStory \rangle, +1), (\langle John, IndianaJones \rangle, +1), (\langle John, Music\&Lyrics \rangle, -1)\}$.

SVM learning from such a dataset can be simply conceived as operating on simple vectors of real numbers representing features of the data points $x^i \in \mathcal{D}$ in a vector space, and in the case of classification, the goal is to find a hyperplane in this space that separates points in $\mathcal{D}^-$ from points in $\mathcal{D}^+$. More formally, given the dataset, the standard SVM algorithm learns a discriminant function $f(x)$:

$$f(x) = w^T \phi(x) + b$$

by solving the following optimization problem:

$$\begin{aligned} \min \quad & \frac{1}{2} \|w\|^2 + C \sum_i \xi_i \\ \text{w.r.t.} \quad & w \in \mathbb{R}^D,\ \xi \in \mathbb{R}^N, b \in \mathbb{R} \\ \text{subject to} \quad & y_i(w^T \phi(x) + b) \geq 1 - \xi_i,\ \xi_i \geq 0 \end{aligned}$$

This optimization is applicable for classification of examples which are not linearly separable. This is where kernels are incorporated into SVM. They can be designed in such a way that implicitly map the input data $\mathcal{D}$ into a higher dimensional space $\mathcal{F}$ where the data points become linearly separable. This mapping is also refer to as the "kernel trick" because it is implicit in the sense that we neither need to access the mapped data, nor the actual mapping function, but it is sufficient to access the results of the inner product of pairs of mapped data points in $\mathcal{F}$. In particular, this explicit mapping is unnecessary given we have a kernel function $k_i(x_1, x_2) = \phi x_1{}^T \phi x_2$ that directly returns the dot product of the mapped data points ϕx_1 and ϕx_2 in some feature space.

The use of a combination of kernels for SVM learning has gained popularity and is particularly promising in the case of semantic data. This kind of data is often sparse in the sense that the feature sets of two given data points might overlap only on some dimensions, e.g. x_1 has an address but x_2 does not. A generic way to focus on some relevant dimensions is to construct an R-convolution kernel introduced by Haussler [6]. Assume that for each data point $x \in \mathcal{X}$ there exists a D-dimensional decomposition $x = (x_1, x_2, .., x_D)$ with each $x_i \in \mathcal{X}_i$, and there are the kernels $k_i : \mathcal{X}_i \times \mathcal{X}_i \rightarrow \mathbb{R}$. Then, Haussler's convolution kernel $K : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ is defined as follows:

$$K(x_1, x_2) = \sum_{x_1' \in R^{-1}(x_1), x_2' \in R^{-1}(x_2)} \mu(x_1', x_2') \prod_{i=1}^{D} k_i(x_1', x_2')$$

where μ denote a set of non-negative coefficients on $\mathcal{X}_i \times \mathcal{X}_i$, and $R : \mathcal{X}_1 \times ... \times \mathcal{X}_D \times \mathcal{X}$ is the decomposition relation. In [6], it is shown that $K(x_1, x_2)$ is positive semi-definite and admissible as a kernel. The intuitive idea of R-convolution is to engineer more sophisticated kernels by tailoring simple and primitive kernels. In this work, we decompose the feature set and learn an R-convolution kernel for every subset. Further, a composite kernel is learned from a non-linear combination of these R-convolution kernels to consider the entire feature set. Note that each of the R-convolution kernels is a composite kernel itself, consisting of sub-kernels that focus on some dimensions.

Learning with Logical Representations. ILP learns logical clauses from data represented using expressive knowledge representation formalisms. A well-known system is FOIL, which we extend in this work for computing relevant features from RDF data. Formally,a *hypothesis* $H = \{c_1, \ldots, c_n\}$ which is a set of clauses where each clause is $h(\overrightarrow{v}) \leftarrow b_1(\overrightarrow{v_1}), .., b_m(\overrightarrow{v_m})$, h and b_i are binary predicates representing the clause's head and body, and $\overrightarrow{v_i}$ are either variables or constants. For example, a clause indicating all users from the *United Kingdom* who likes *comedy* movies can be stated as follows:

$$c_1 : \mathit{likes}(?\mathit{user}, ?\mathit{movie}) \leftarrow \mathit{location}(?\mathit{user}, UK), \mathit{genre}(?\mathit{movie}, \mathit{comedy})$$

A relation "$\preceq$" is called quasi-order on a set $\mathcal{H}$ if it is reflexive (i.e. $a \preceq a$ for all $a \in \mathcal{H}$) and transitive (i.e. $a \preceq b$ and $b \preceq c$ implies $a \preceq c$). Given a search space $(\mathcal{H},\preceq)$ of a possible clauses and hypotheses, respectively, a *refinement operator* ρ implies a mapping of a clause c to a set of clauses $\rho(c)$, such that $\rho(c) = \{c' \mid c' \in \mathcal{H}, c' \preceq c\}$. Computing the hypothesis is based on searching and refining clauses in this space. The goal is to find a hypothesis that "covers" all positive examples and does not cover negative examples. While different cover relational have been used, learning from entailment is most widely applied, i.e., the coverage relational $covers(H, x)$ returns true iff $H \models x$ such that the result of the learning is the set $\{H | \forall x_1 \in \mathcal{D}^+ : H \models x_1 \wedge \forall x_2 \in \mathcal{D}^- : H \not\models x_2\}$.

3 Learning from RDF Graphs

In this section, we propose a Relational Kernel Machine for learning from graph-structured data, that can be applied to RDF data. First, we show how the ILP approach for finding hypothesis can be employed to obtain a R-convolution kernel that is a combination of clause kernels. We also present the notion of clause kernel and discuss how the computation of the hypothesis and kernel can be formulated as a multi-objective optimization problem, and finally explain the use of a genetic algorithm to solve this.

3.1 Clause Kernel

Observe that ILP aims to a find a hypothesis that capture the structure and semantics of a given set of examples. It can be seen as an intensional description of the example feature set. Further, the constituent clauses actually define how this feature set can be decomposed into subsets. This ability of the hypotheses to define features as graph substructures is quite useful for constructing a composite kernel from clause kernels, i.e., kernels constructed for every clause. In turn, as every clause contains a set of predicates, which stand for dimensions of a feature subset, we propose to construct a R-convolution kernel for every clause to employ simple kernels on dimensions.

We define a clause kernel $K_c(x_1, x_2)$ over each pair of data points $x_1, x_2 \in X$. Following the R-convolution kernels [6], we define a decomposition relation $R_c^{-1}(x)$ over the data point x for a clause c as follows:

- Given $x = \langle x_1, x_2 \rangle$, a substitution $\theta = \{V_1/x_1, V_2/x_2\}$ instantiates the variables V_1, V_2 of c with x as $c\theta$, and
- given that instantiated clause without the head predicate denoted $c_{body}\theta$, another substitution $\theta' = \{V_3/b_3, \cdots, V_m/b_m\}$ instantiates the remaining variables V_i in $c_{body}\theta$ with bindings b_i as $c_{body}\theta\theta'$,
- then θ' is a R-decomposition relation for c denoted $R_c^{-1}(x)$ iff $G \models c_i\theta\theta'$ where G denotes the data graph.

In the RDF context, entailment is evaluated simply via graph pattern matching, i.e., $G \models c_i\theta\theta'$ if $c_i\theta\theta'$ is a binding to the query pattern $c_{body}\theta$ such that it is a subgraph of G.

Intuitively speaking, this definition refers to the grounding of some variables in a clause with the data x. For example, for a clause as:

$$c_1 : likes(?user, ?movie) \leftarrow location(?user, ?c), age(?user, ?a), genre(?movie, ?g)$$

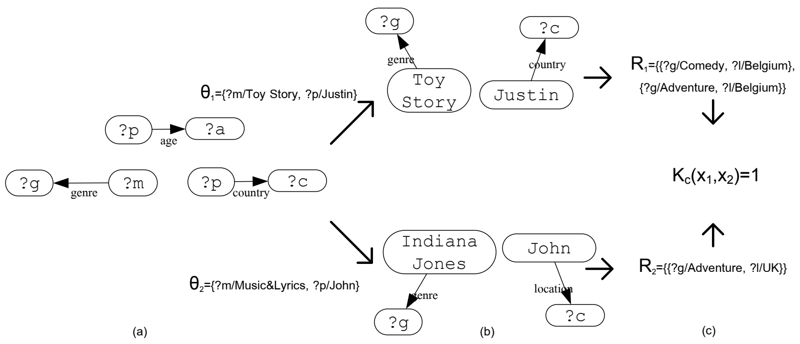

?p
?a
age
?g
?m
genre
?p
?c
country
θ1={?m/Toy Story, ?p/Justin}
θ2={?m/Music&Lyrics, ?p/John}
?g
genre
Toy Story
Justin
country
?c
Indiana Jones
genre
?g
John
location
?c
R1={{?g/Comedy, ?l/Belgium}, {?g/Adventure, ?l/Belgium}}
Kc(x1,x2)=1
R2={{?g/Adventure, ?l/UK}}
(a)
(b)
(c)

Fig. 2. An example calculation of clause kernel: (a) graph representation of a clause without head, (b) instantiated with data, (c) the results of the instantiated clauses evaluated as graph patterns and the resulting kernel value.

a substitution $\theta = \{?user/John, ?movie/Inception\}$ for $x = \langle John, Inception\rangle$ results in the following instantiated clause:

$$p_1 : likes(John, Inception) \leftarrow location(John, ?c), age(John, ?a), genre(Inception, ?g)$$

A kernel over two data points (x_1,x_2) then can be calculated by instantiating c to obtain $c\theta_1$ and $c\theta_2$, and showing how similar x_1 and x_2 are based on these instantiated clauses. A trivial evaluation of this similarity is to consider the instantiated clauses without their heads $c'_i\theta$ as graph patterns to be evaluated on the data graph. For this, given G, we define a result set $S_{c'\theta} = \{\langle r_1, .., r_m\rangle \mid \theta' = \{V_3/r_3, .., V_m/r_m\} \wedge G \models c'\theta\theta'\}$. Based on two result sets, a kernel function can be defined as:

$$k(c'_1\theta, c'_2\theta) = \mid \{\langle r_i, r_j\rangle \mid r_i \in S_{c'_1\theta} \wedge r_j \in S_{c'_2\theta} \wedge r_i = r_j\} \mid$$

Intuitively speaking, given $c'\theta$ as a set of feature dimensions, the similarities of two data points are measured by retrieving values of those dimensions represented by variables in $c'\theta$, and check if these points agree on these values. The resulting clause kernel is a R-convolution kernel in the sense that $R_c^{-1}(x)$ decomposes the feature set captured by c into dimensions, and subkernels k_i are used to operate on these dimensions, i.e., $k_i(x'_1, x'_2)$ where $x'_1 \in R_c^{-1}(x_1)$ and $x'_2 \in R_c^{-1}(x_2)$. An example illustrating this is given in Fig. 2. In this case, the two data points agree only on the value *Adventure* for the dimension *genre*.

3.2 Kernel Learning

The kernel function we aim to learn here is constructed as a non-linear combination of many small clause kernels, which in turn, can be further decomposed into dimensions as discussed. As discussed previously, the basic clause kernels can be indexed by a set H that corresponds to a hypothesis in our case. Therefore, we adopt a formulation of kernel learning based on the search of clauses over the search space $(\mathcal{H},\preceq)$ where $\mathcal{H}$ indicates the set of all possible clauses and hypotheses, respectively. Then, we propose the following optimization scheme:

$$\begin{aligned} &\inf_{H\subset\mathcal{H}} \min_{w,\xi,b} \quad \frac{1}{2}\sum_{c\in H}\frac{1}{d_c}\left\|w_c\right\|^2 + C\sum_i \xi_i \\ &\text{subject to} \quad y_i(\sum_{c\in H} w_c^T\phi_c(x)+b) \geq 1-\xi_i,\ \sum_{c\in H} d_c = 1,\ \xi_i \geq 0,\ d_c \geq 0 \end{aligned} \quad (1)$$

In the formulation above, on the one hand, the inner minimization problem represents a multiple kernel learning setting, where each clause c in a given hypothesis H contributes to the solution controlled by the value d_c. Thus, each d_c controls the importance given to the squared norm of w_c in the objective function. In [10], this inner optimization is solved as a 2-step alternating optimization algorithm, where the first step consists of solving w, b, and ξ for a fixed d, and the second step consists of solving the vector d for fixed w, b, and ξ. On the other hand, the outer infimum problem represents a hypothesis search where a set of clauses H are determined over a hypothesis search space $\mathcal{H}$. Introducing Lagrange multipliers α, λ, we derive the following dual problem for optimization with kernels K_c:

$$\begin{aligned} &\sup_{H\subset\mathcal{H}} \max_{\alpha} \quad \sum_i \alpha_i - \lambda \\ &\text{subject to} \quad \sum_i \alpha_i y_i = 0,\ 0\leq\alpha_i\leq C\ i=1,...,n, \\ &\qquad F(H,d)\leq\lambda\ \forall.H\in 2^{\mathcal{H}} \end{aligned} \quad (2)$$

where $F(H,d)$ is defined as:

$$F(H,d) = \frac{1}{2}\sum_{i,j}\alpha_i\alpha_i y_i y_j \sum_{c\in H} d_c K_c(x_i,x_j) \quad (3)$$

The solution to the problem above can actually be decomposed into two parts: First, by directly solving the inner maximization, we obtain the optimal points (α^*,λ^*) for dual variables. Then, plugging the optimal point α^* into $F(H,d)$, the outer optimization problem yields to the following equivalent optimization:

$$\min_{H\in 2^{\mathcal{H}},d} F(H,d) = \frac{1}{2}\sum_{i,j}\alpha_i^*\alpha_i^* y_i y_j \sum_{c\in H} d_c K_c(x_i,x_j) \quad (4)$$

Such an optimization, however, requires the hypothesis to be pre-given before solving the kernel learning problem. For solving this optimization problem, we also apply a refinement operator that changes the hypothesis iteratively. It is initiated with a top-level clause, indicating the start of the hypothesis and refines it in each step before solving the two-part optimization. At each step, a *refine* function is called that executes a refinement operator $\rho(c')$ over a selected clause c' from the current hypothesis H^t. Refinement operator performs the refinement in two ways:

1. It picks a predicate $b_i(\overrightarrow{v_i})$ from the selected clause c' and finds another predicate b_j that is connected to b_i in the RDF graph by a shared node. For example, for the RDF graph shown in Figure 1, a predicate $location(?user, ?l)$ can be extended by $age(?user, ?a)$.
2. It replaces a free variable in the clause with a ground term (i.e. URI) of an entity. In order to select the best grounded term, it selects the most-used grounded term in the substitutions θ of clause kernels. For instance, the term *Adventure* is used to ground the variable $?g$ in the predicate $genre(?m, ?g)$ according to the example presented in subsection 3.1.

A refinement operator computes a set of refined clauses $\{c_1, .., c_k\}$ in each refinement. In order to select the best clause to include into the hypothesis, we apply Kernel Target Alignment (KTA) [2] to the new clauses to determine how each clause kernel K_{c_i} is aligned with the ideal kernel Y. Note that ideal kernel Y is calculated over the labels in the dataset with an outer product $Y = yy^T$, where $y = (y_1, ..., y_n)^T$. A simple measure of the alignment is provided between the clause kernel K_{c_i} and Y by a Frobenius inner product $< ., . >_F$ as:

$$< Y, K_{c_i} >_F = \sum_{y_i = y_j} K_{c_i}(x_i, x_j) - \sum_{y_i \neq y_j} K_{c_i}(x_i, x_j) \tag{5}$$

This type of refinement is similar to the ILP techniques such as FOIL in terms of dynamically inducing new clauses of the hypothesis [9]. However, it differs in the way we utilize RDF graph to extend the clauses and test the alignment over the kernel functions before accepting a clause into the hypothesis.

3.3 Algorithm

In this section we present the overall algorithm that solves the hypothesis refinement and kernel learning together. Algorithm 1 shows the overall process which starts with a given initial target clause. To construct the first hypothesis, the *refine* function is called once as depicted in Algorithm 2. Mainly, this function selects a clause c' from a given hypothesis, applies the refinement operator $\rho(c')$ to obtain a set of refined clauses, and selects a subset of these refined clauses by using the KTA to be replaced in the hypothesis with the selected clause c'. The parameter θ indicates the number of clauses to be replaced which we set to 2 in our experiments.

Algorithm 1, then, continues with two nested loops: In the inner loop, kernel learning takes place as a two-step process. A SVM solver is used to obtain optimal (α^*, λ^*) and they are used to update the weight vector $\mathbf{d}$. For this, we take a simple differentiation of the dual function F w.r.t. every weight d_c as $\frac{\partial F}{\partial d_c} = \frac{1}{2}\sum_{i,j=1}^{n} \alpha_i^* \alpha_j^* y_i y_j K_c(x_i, x_j)$ and apply an updating scheme such as $d_c \leftarrow d_c + \gamma \frac{\partial F}{\partial d_c}$, where γ is the step size chosen according to Armijo rule as suggested in [10]. In the outer loop, the *refine* function is called again to change the hypothesis and the optimization is done again in the next iteration.

Algorithm 1. Kernel Learning Algorithim

```
Input: $C^0$:initial clause
  $t \leftarrow 0$
  $(H^0, \mathbf{d}^0) \leftarrow refine(\{C^0\}, \{1\})$
  repeat
    repeat
      $K_H \leftarrow \sum_{c \in H^t} d_c^t K_c$
      $(\alpha^*, \lambda^*) \leftarrow$ Solve the SVM problem in Eq. 2 for $K_H$
      $S \leftarrow F(H^t, \mathbf{d}^t)$
      $d_c^t \leftarrow d_c^t + \gamma \frac{\partial F}{\partial d_c}$, where $\forall c \in H^t$ and for $\alpha^*$
    until $F(H^t, \mathbf{d}^t) \geq S$
    $(H^{t+1}, \mathbf{d}^{t+1}) \leftarrow refine(H^t, \mathbf{d}^t)$
    $t \leftarrow t + 1$
  until converge
```

Algorithm 2. Refine

```
Input: $H^t$: Hypothesis to be refined, $\mathbf{d}^t$: Weight vector
  Initialize $H^{t+1} \leftarrow H^t$, $\mathbf{d}^{t+1} \leftarrow \mathbf{d}^t$
  $c' \leftarrow$ Select a clause from $H^t$
  Apply refinement operator $\{c_1, ..., c_k\} \leftarrow \rho(c')$
  $mina \leftarrow -\infty$, $minI \leftarrow -1$
  $newClauses = \emptyset$
  for $i = 1$ to $k$ do
    $a_i \leftarrow$ Calculate alignment of $c_i$ in Eq. 5
    if $a_i > mina$ then
      $newClauses \leftarrow newClauses \cup c_i$
      if $| newClauses | > \theta$ then
        Remove $c_{minI}$ from $newClauses$
      end if
      $mina \leftarrow a_i$, $minI \leftarrow i$
    end if
  end for
  for all $c \in newClauses$ do
    $H^{t+1} \leftarrow H^t \cup c$
    $d_c^{t+1} \leftarrow \frac{d_{c'}^t}{|newClauses|}$
  end for
  Remove $c'$ from $H^{t+1}$ and $d_{c'}$ from $\mathbf{d}^{t+1}$
  return $H^{t+1}$, $\mathbf{d}^{t+1}$
```

Algorithm 1 iteratively refines the hypothesis and trains a SVM in each iteration. Clearly, this is an expensive procedure because optimal parameters have to be computed for all possible candidate hypothesis, even though many of them may turn out to be non-optimal, i.e., yield high prediction error. We approximate this process by means of co-evolutionary genetic algorithm. Basically, GA iteratively applies crossover and mutation on the fittest solutions to breed new refined solutions. Two different but dependent species of individuals constituting two subpopulations are to be trained. The first subpopulation consists of individuals representing hypotheses and the second one is meant to train the SVM coefficients. It should be noted that the two individuals combined from both subpopulations form one candidate solution to our nested two-loop algorithm. Dual objective is used as a fitness function. Crossover and mutation is applied to two subpopulations separately as the individuals representing a hypothesis is refined in each generation with the *refine* function, and individuals representing SVM coefficients are randomly changed with a uniform crossover and two-point mutation.

3.4 Relational Kernel Machines

The purpose of the learner is to find the coefficients and a multiple kernel for a prediction function which should have the following form:

$$f(x) = \sum_i \alpha_i y_i \sum_{c \in H} d_c K_c(x_i, x)$$

This function can be constructed by selecting two individuals from each one of the subpopulations resulting from our coevolutionary optimization. After the optimization is finished, we can use the prediction error for each pairs of individuals selected to obtain the best solution to the classification problem. Different variations of loss functions can be applied here. We use the hinge loss function:

$$l(y, f(x)) = max(0, 1 - y * f(x))$$

4 Experiments

To demonstrate the general feasibility of our learning approach, we conducted experiments on two different evaluation scenarios. In this section, we present each setting, the extracted data-sets and discussion on the results. Our implementation, RDFLearner, is publicly available at http://code.google.com/p/rdflearner/.

4.1 The SWRC Ontology

In the first experiment we compare our approach with related work on kernel-based SVM learning [1].

Baseline. In particular, this approach proposes the use of a number of predefined kernels in Support Vector Machines, such as SVMlight[1]. The kernels used in the previous experiment [1] representing the baseline of this work are: *CCOP1*: A combination of common class similarity kernel and two object property kernels on *workedOnBy* and *worksAtProject*; *CCOP2*: The same as CCOP1 with the property *publication* in addition; and *CCOPDP*: CCOP2 plus a data property kernel on *title* of publications. The result was obtained by summing up the combination of corresponding weighted kernels, where the weights were manually defined and set to the same values as used before [1].

[1] `http://svmlight.joachims.org/`

Data. For comparison purpose, we use the classification scenario and dataset that was employed previously [1]. As data, we have the SWRC Ontology[2], which contain data from the Semantic portal of the Institute AIFB. This ontology provides a vocabulary for modeling entities and their relations in the research environment. Top-level concepts include *Person, Publication, Event, Organization, Topic* and *Project*. The associated data contain 178 instances of type *Person* that have an affiliation to one of the 4 institute's research groups. 78 of them are employed as research staff. There are 1232 *Publication* instances, 146 instances of the type *ResearchTopic* and 146 *Project* instances.

Classification Scenario. The task of the SVMs is to predict the affiliation of a person. We take the people in the data, which are associated to the each research group, as positive examples and train a binary classifier for each group. This means we have 4 classifiers which predict whether a person belongs to a particular group or not. The final result is the average of the 4 prediction results. For the experiments the Leave-One-Out cross-validation strategy with 1 as the soft margin parameter for the SVM was applied. It means that for every classifier one example is left out and the classifier is trained over the other examples and tested on the selected one. This is repeated for every example.

Table 1. Person2Affiliation experiment results

Kernel	Error	Precision	Recall	F
CCOP1	6.88	85.32	46.07	59.83
CCOP2	6.04	90.70	48.78	63.44
CCOPDP	4.49	95.87	57.27	71.71
RDFLearner1	0.85	99.26	97.12	98.18
RDFLearner2	28.25	95.83	31.94	47.91

We use two configurations of our approach called *RDFLearner1* and *RDFLearner2*. For both, the optimization was performed with refinement based on 100 generations and 30 individuals in each subpopulation. We use the function measuring recall as the scoring function.

RDFLearner1 was learned from the full initial dataset. Its results in comparison to the baseline kernels are shown in Table 1. Due to dynamic induction of features, it found for instance *distinguishing features such as* that those *Person* instances that were selected for the positive examples were also instances of type *Employee*. This feature was found by the learner almost in all cases. With the help of dynamic weighting, RDFLearner1 was able to give more importance to that feature relative to others, resulting in higher precision and recall as shown in Table 1.

For RDFLearner2, we deliberately excluded two features from the dataset, namely the *Employee* class membership and *affiliation* relation. These are crucial features which have strong impact in the SVM learned by RDFLearner1.

[2] `http://ontoware.org/swrc/`

Omitting these, the results of *RDFlearner2* were considerably poorer, recall in particular. However, by adopting a different combination of features and weights for the modified dataset, *RDFlearner2* still provide relatively high precision, as shown in Table 1.

Thus, the prediction quality offered by RDFLearner is good, and RDFLearner outperformed the baseline when the complete dataset was used. Still, we like to emphasize the major difference of the two approaches is that the baseline classifier [1] uses predefined features and weights. This method is simple and can show good results but is applicable only when sufficient domain and expert knowledge is available. In fact, these weights and kernels have to be defined and optimized manually for every classification scenario. Both features and weights are learned automatically in our approach.

4.2 Context-Aware Movie Recommendation

As a second experiment, we provide a more comprehensive evalation of RDFLearner on a scenario of context-aware movie recommendations and our experiments are based on the anonymized Filmtipset[3] dataset. It contains information about the user ratings of movies, the time of the ratings, the people starring in movies, directors, user profile information like home country, age, gender, and movie specific information like genre, comments etc. We randomly select 150 users (out of 1000) from this dataset and the movie ratings (in our case *likes* or *dislikes*, which indicate the positive and negative examples), the movie information, location, and user profile information. This subset is divided into a training set with around 80 percent and a test set with around 20 percent of the examples. We convert these data to obtain a RDF representation.

We are interested in predicting the predicate $like(?user, ?movie)$.

Accuracy. First, we focus on the prediction performance under different parameter settings. One way to measure this is using the accuracy defined as $ACC = \frac{(TP+TN)}{(P+N)}$ where TP are true positives, TN - true negatives, P - all positives and N - all negatives. Thus, prediction result has high accuracy when it contains few false positives and false negatives. Figure 3-a plots the accuracy values in percent against different settings we used for the co-evolution, i.e., for different numbers of generations and population sizes. We can observe that accuracy increased both with the number of generations and population size. However, at a certain point, incrementing these numbers does not yield much improvement. We consider this point to be domain and dataset specific and should be found out with experimental observations.

Accuracy Compared to Standard SVM. In addition, at each iteration of the optimization algorithm, we replace the genetic refinement and mutation with a standard SVM classifier for calculating the SVM coefficients. Against this baseline, we compared prediction quality as well as running time. The configuration we used for the following experiments is: 100 generations and 50 population size.

[3] `http://www.filmtipset.se/`

The results shown in Figure 3-b indicate that accuracy of both approaches increased with the size of the dataset. We noticed that they generated different coefficients and different numbers of support vectors. Standard SVM aims to find a global optimum of the data points, resulting in a smaller number of support vectors compared to our approach. Despites this, accuracy results of both approaches were quite similar, indicating that our approach was good at finding optimized coefficients.

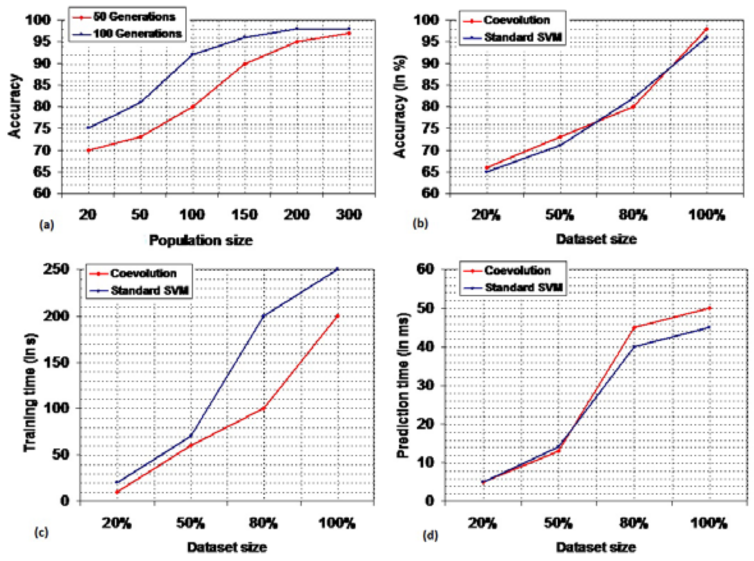


Fig. 3. Experiment Results regarding a) Accuracy against generation and population size, b) accurracy with co-evolution and SVM training of coefficients, c) training time, and d) prediction time

Running Time Compared to Standard SVM. However, differences in these approaches take influence on the running time. We distinguished the running time of training from running time of prediction. The training time of RDFLearner and the SVM baseline can be observed in Figure 3-c. These results clearly show that applying the heuristics in the co-evolution has significant improvements over standard SVM training. This effect is pronounced when the dataset is large. However, standard SVM is slightly faster at prediction, as shown in Figure 3-d. This is due to the smaller number of support vectors used by the SVM, which means less complex computation is required for prediction.

5 Related Work

The use of learning methods on semantic data receives attention in recent years. Approaches presented in [4,3,1] employ kernels to learn from semantic data by using state-of-the-art techniques. Mainly, these approaches discuss relevant features in semantic data, and based on them, specify the kernels to be used by the classifier. In comparison to these approaches, we provide the most generic notion of kernel based on clauses that might capture arbitrary structures (and features). Features and their weights to be used in the kernel are learned and optimize automatically.

Another related field of research involves the use of SRL techniques [7,11]. In [7] the authors propose to extend SPARQL with mining support based on the use of existing SRL approaches such that both deductive and inductive reasoning can be employed for query answering. Also, in [11], it has been shown how ontological information can be used to train a probabilistic model with prior knowledge. The drawback of SRL approaches in general is the need to construct a graphical model (e.g. Bayesian or Markov network). This is costly and might not be affordable when dealing with large networks like semantic data on the Web.

In this regard, using kernels in the ILP setting has been shown to perform well, especially when the dimensions of the feature space are unknown. A comprehensive survey discussing the relationship between the kernel-based techniques and ILP is presented in [5]. One prominent example that combines kernel methods and ILP is kFOIL, which utilizes kernel-based learning with hypothesis induction [8]. Though it is conceptually similar to our approach in the sense of using a combination of logical learning and kernel methods, it has been applied to the problem of ILP whereas we use this combination for SVM learning. Further, the applicability of this approach is limited in the semantic data setting, both in terms of the employed kernel and efficiency of optimization. In particular, kFOIL employs a kernel based on logical entailment whereas we compute similarities by considering the surrounding of nodes in the data graph (as captured by a clause). In addition, kFOIL employs an iterative algorithm that revises the hypothesis and trains an SVM in each iteration – this is recognized to be highly inefficient [8]. In our approach, we solve this issue by mapping the optimization problem to kernel learning and utilize a coevolutionary algorithm that searches hypotheses and SVM coefficients simultaneously.

6 Conclusion

We presented the concept of Relational Kernel Machines which is able to automatically identify and optimize features and their weights for learning kernels and a SVM from graph-structured semantic data. It employs ILP-based dynamic propositionalization to compute relevant features by searching for a hypothesis which serves as a description of features. This description is further decomposed into clauses, for which we learn R-convolution kernels, each can be seen as a composite kernel with sub-kernels focusing on some feature dimensions captured by the clauses. In turn, these clause kernels are combined and used as the basis for our kernel machine. The resulting problem is basically multi-objective optimization, as hypothesis and the kernel learning are dependent processes that have to be considered simultaneously. We propose the use of a coevolution algorithm to deal with this. Besides the fact that our approach does not require expert knowledge for finding and tuning features and weights, the experiments showed that our approach outperformed the baseline SVM approach which relies on manually defined kernels.

References

1. Bloehdorn, S., Sure, Y.: Kernel methods for mining instance data in ontologies. In: Aberer, K., Choi, K.-S., Noy, N., Allemang, D., Lee, K.-I., Nixon, L.J.B., Golbeck, J., Mika, P., Maynard, D., Mizoguchi, R., Schreiber, G., Cudré-Mauroux, P. (eds.) ASWC 2007 and ISWC 2007. LNCS, vol. 4825, pp. 58–71. Springer, Heidelberg (2007)
2. Cristianini, N., Kandola, J., Elisseeff, A., Shawe-Taylor, J.: On kernel target alignment. Innovations in Machine Learning, 205–256 (2006)
3. d'Amato, C., Fanizzi, N., Esposito, F.: Classification and retrieval through semantic kernels. In: Lovrek, I., Howlett, R.J., Jain, L.C. (eds.) KES 2008, Part III. LNCS (LNAI), vol. 5179, pp. 252–259. Springer, Heidelberg (2008)
4. Fanizzi, N., d'Amato, C., Esposito, F.: Learning with kernels in description logics. In: Železný, F., Lavrač, N. (eds.) ILP 2008. LNCS (LNAI), vol. 5194, pp. 210–225. Springer, Heidelberg (2008)
5. Frasconi, P., Passerini, A.: Learning with kernels and logical representations. In: De Raedt, L., Frasconi, P., Kersting, K., Muggleton, S.H. (eds.) Probabilistic Inductive Logic Programming. LNCS (LNAI), vol. 4911, pp. 56–91. Springer, Heidelberg (2008)
6. Haussler, D.: Convolution kernels on discrete structures (Technical Report UCSC-CRL-99-10). University of California, Santa Cruz (1999)
7. Kiefer, C., Bernstein, A., Locher, A.: Adding data mining support to SPARQL via statistical relational learning methods. In: Bechhofer, S., Hauswirth, M., Hoffmann, J., Koubarakis, M. (eds.) ESWC 2008. LNCS, vol. 5021, pp. 478–492. Springer, Heidelberg (2008)
8. Landwehr, N., Passerini, A., De Raedt, L., Frasconi, P.: kFOIL: Learning simple relational kernels. In: Proceedings of the National Conference on Artificial Intelligence, vol. 21, p. 389. AAAI Press, Menlo Park (1999), MIT Press, Cambridge (2006)
9. Quinlan, J.: Learning logical definitions from relations. Machine Learning 5(3), 239–266 (1990)
10. Rakotomamonjy, A., Bach, F., Canu, S., Grandvalet, Y.: More efficiency in multiple kernel learning. In: Proceedings of the 24th International Conference on Machine Learning, pp. 775–782. ACM, New York (2007)
11. Rettinger, A., Nickles, M., Tresp, V.: Statistical relational learning with formal ontologies. In: Buntine, W., Grobelnik, M., Mladenić, D., Shawe-Taylor, J. (eds.) ECML PKDD 2009. LNCS, vol. 5782, pp. 286–301. Springer, Heidelberg (2009)

AutoSPARQL: Let Users Query Your Knowledge Base

Jens Lehmann and Lorenz Bühmann

AKSW Group
Department of Computer Science
Johannisgasse 26
04103 Leipzig, Germany
{lehmann,buehmann}@informatik.uni-leipzig.de

Abstract. An advantage of Semantic Web standards like RDF and OWL is their flexibility in modifying the structure of a knowledge base. To turn this flexibility into a practical advantage, it is of high importance to have tools and methods, which offer similar flexibility in exploring information in a knowledge base. This is closely related to the ability to easily formulate queries over those knowledge bases. We explain benefits and drawbacks of existing techniques in achieving this goal and then present the QTL algorithm, which fills a gap in research and practice. It uses supervised machine learning and allows users to ask queries without knowing the schema of the underlying knowledge base beforehand and without expertise in the SPARQL query language. We then present the AutoSPARQL user interface, which implements an active learning approach on top of QTL. Finally, we evaluate the approach based on a benchmark data set for question answering over Linked Data.

1 Introduction and Motivation

The Web of Data has been growing continuously over the past years and now contains more than 100 public SPARQL endpoints with dozens of billion of triples.[1] The data available via those endpoints spans several domains reaching from music, art and science to spatial and encyclopaedic information as can be observed at `http://lod-cloud.net`. Providing this knowledge and improving its quality are important steps towards realising the Semantic Web vision. However, for this vision to become reality, knowledge also needs to be easy to query and use.

Typically, querying an RDF knowledge base via SPARQL queries is not considered an end user task as it requires familiarity with its syntax and the structure of the underlying knowledge base. For this reason, query interfaces are often tight to a specific knowledge base. More flexible techniques include facet based browsing and graphical query builders. We briefly analyse advantages and disadvantages of those approaches and explain how they relate to AutoSPARQL.

[1] See `http://ckan.net` for data set statistics.

© Springer-Verlag Berlin Heidelberg 2011

Knowledge Base Specific Interfaces: Special purpose interfaces are often convenient to use since they usually shield the user from the complexity and heterogeneity of the underlying knowledge bases. The vast majority of web search forms fall into this category. Such interfaces are often designed to capture the most relevant queries users may ask. However, those queries have to be known in advance. They usually do not allow to explore the underlying RDF graph structure. Other disadvantages are the development effort required for developing specific interfaces and their inflexibility in case of schema changes or extensions. *Facet-Based Browsing* is a successful technique for exploring knowledge bases, where users are offered useful restrictions (facets) to the resources s/he is viewing. The technique is not knowledge base specific and, thus, requires no or only small adaptations to be used on top of existing SPARQL endpoints. Two examples are the Neofonie Browser `http://dbpedia.neofonie.de`, the Virtuoso facet service `http://dbpedia.org/fct/` and OntoWiki[2] facets. The first is tailored towards DBpedia, whereas the latter two examples can be run on top of arbitrary knowledge bases. A disadvantage of facet-based browsers is that they allow only a limited set of queries. For instance, it is easy to query for objects belonging to a class "Person". However, facets do not work well for more complex queries like "Persons who went to school in Germany", because the restriction "in Germany" refers to the school and not directly a person. Another type of difficult queries is "Persons who live in x", where x is a small city. In this case, the difficulty is that the facet "live in x" may not be offered to the user, because there are many other more frequently occurring patterns offered as facets.

Visual SPARQL Query Builders lower the difficulty of creating SPARQL queries. However, their target user groups are still mostly knowledge engineers and developers. Examples of visual query builders are SPARQL Views[3] [4] and Virtuoso Interactive Query Builder[4]. Even though the queries are visualised, users still need some understanding of how SPARQL queries actually work and which constructs should be used to formulate a query. To visually build a query, users also need a rough understanding of the underlying schema.

Question Answering (QA) Systems allow the user to directly enter his question, e.g. in Ginseng[5], NLP-Reduce[6] or PowerAqua[7]. Usually, they need to be adapted to a particular domain, e.g. via patterns or models. Cross domain question answering without user feedback can be brittle.

AutoSPARQL: In this paper, we propose the QTL algorithm and the AutoSPARQL user interface. It provides an alternative to the above interfaces with a different set of strengths and restrictions. AutoSPARQL uses active supervised

[2] `http://ontowiki.net`

[3] `http://drupal.org/project/sparql_views`

[4] `http://wikis.openlinksw.com/dataspace/owiki/wiki/OATWikiWeb/InteractiveSparqlQueryBuilder`

[5] `http://www.ifi.uzh.ch/ddis/research/talking-to-the-semantic-web/ginseng/`

[6] `http://www.ifi.uzh.ch/ddis/research/talking-to-the-semantic-web/nlpreduce/`

[7] `http://technologies.kmi.open.ac.uk/poweraqua/`

machine learning to generate a SPARQL query based on positive examples, i.e. resources which should be in the result set of the SPARQL query, and negative examples, i.e. resources which should not be in the result set of the query. The user can either start with a question as in other QA systems or by directly searching for a relevant resource, e.g. "Berlin". He then selects an appropriate result, which becomes the first positive example. After that, he is asked a series of questions on whether a resource, e.g. "Paris", should also be contained in the result set. These questions are answered by "yes" or "no". This feedback allows the supervised learning method to gradually learn which query the user is likely interested in. The user can always observe the result of the currently learned query and stop answering questions if the algorithm has correctly learned it. The system can also inform the user if there is no learnable query, which does not contradict with the selection of positive and negative examples. AutoSPARQL can generate more complex queries than facet based browsers and most knowledge base specific applications, while there are some restrictions – explained in detail later – compared to manually or visually creating queries. We argue that it is easier to use than manual or visual SPARQL query builders and not much more difficult to use than facet-based browsers or standard QA systems. Due to this different sets of strengths and weaknesses, it provides a viable alternative to the methods described above. Our claim is that AutoSPARQL is the first user interface, which allows end users to create and refine non-trivial SPARQL queries over arbitrary knowledge bases.

Overall, we make the following contributions:

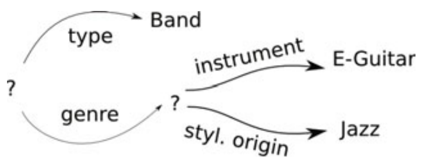
– introduction of a new active learning method for creating SPARQL queries
– the Query Tree Learner (QTL) algorithm
– the AutoSPARQL interface at `http://autosparql.dl-learner.org`
– an evaluation on a benchmark data set for question answering over Linked Data

Sections 2 to 4 are the formal part of the paper. In Section 2, we introduce the concept of query trees as underlying structure. After that, we show basic operations on those trees and proof their properties in Section 3. The following section explains the QTL algorithm, which combines results from the previous sections. In Section 5, the AutoSPARQL workflow and user interface are presented. In Section 6, we measure how well AutoSPARQL works on a benchmark data set for question answering over Linked Data. Related work is described in Section 7. Finally, some high level key aspects of our approach are discussed in Section 8.

2 Query Trees

Before explaining query trees, we fix some preliminaries. We will often use standard notions from the RDF and SPARQL specifications[8], e.g. triple, RDF graph,

[8] `http://www.w3.org/TR/rdf-concepts`,
`http://www.w3.org/TR/rdf-sparql-query`



```
SELECT ?x0 WHERE {
?x0 rdf:type dbo:Band.
?x0 dbo:genre ?x1.
?x1 dbo:instrument dbp:Electric_guitar.
?x1 dbo:stylisticOrigin dbp:Jazz.
}
```

Fig. 1. Query tree (left) and corresponding SPARQL query (right)

triple pattern, basic graph pattern. We denote the set of RDF resources with R, the set of RDF literals with L, the set of SPARQL queries with SQ, and the set of strings with S. We use $f_{|D}$ to denote the restriction of a function to a domain D.

We call the structure, which is used internally by the QTL algorithm, a *query tree*. A query tree roughly corresponds to a SPARQL query, but not all SPARQL queries can be expressed as query trees.

Definition 1 (Query Tree). *A* query tree *is a rooted, directed, labelled tree $T = (V, E, \ell)$, where V is a finite set of* nodes*, $E \subset V \times R \times V$ is a finite set of* edges*, $NL = L \cup R \cup \{?\}$ is a set of* node labels *and $\ell : V \to NL$ is the* labelling function*. The* root *of T is denoted as $root(T)$. If $\ell(root(T)) = ?$, we call the query tree* complete*. The set of all query trees is denoted by $\mathcal{T}$ and $\mathcal{T}_C$ for complete query trees. We use the notions $V(T) := V$, $E(T) := E$, $\ell(T) := \ell$ to refer to nodes, edges and label function of a tree T. We say $v_1 \xrightarrow{e_1} \cdots \xrightarrow{e_n} v_{n+1}$ is a* path *of length n from v_1 to v_{n+1} in T iff $(v_i, e_i, v_{i+1}) \in E$ for $1 \le i \le n$. The depth of a tree is length of its longest path.*

Definition 2 (Subtrees as Query Trees). *If $T = (V, E, \ell)$ is a query tree and $v \in V$, then we denote by $T(v)$ the tree $T' = (V', E', \ell')$ with $root(T') = v$, $V' = \{v' \mid \text{there is a path from } v \text{ to } v'\}$, $E' = E \cap V' \times R \times V'$ and $\ell' = \ell_{|V'}$.*

Definition 3 (Node Label Replacement). *Given a query tree T, a node $v \in V(T)$ and $n \in NL$, we define $\ell[v \mapsto n]$ as $\ell[v \mapsto n](v) := n$ and $l[v \mapsto L](w) := \ell(w)$ for all $w \neq v$. We define $T[v \mapsto n] := (V(T), E(T), \ell(T)[v \mapsto n])$ for $v \in V(T)$. We say that the* label of v is replaced by n.

2.1 Mapping Query Trees to SPARQL Queries

Each query tree can be transformed to a SPARQL query. The result of the query always has a single column. This column is created via the label of the root node, which we will also refer to as the *projection variable* in this article. Please note that while QTL itself learns queries with a single column, i.e. lists of resources, those can be extended via the AutoSPARQL user interface. Each edge in the query tree corresponds to a SPARQL triple pattern. Starting from the root node, the tree is traversed as long as we encounter variable symbols. Each variable symbol is represented by a new variable in the SPARQL query. An example is shown in Figure 1.

Formally, the mapping is defined as follows: Each node with label ? is assigned a unique number via the function id : $V \mapsto \mathbb{N}$. A root node is assigned value 0. The function mapnode : $V \mapsto S$ is defined as: $mapnode(n) =$ "$?x$" $+ \text{id}(n)$ if $\ell(n) =?$ and $mapnode(n) = n$ otherwise. Note that "+" denotes string concatenation. Based on this, the function mapedge : $E \mapsto S$ is defined as $map(v) +$ " " $+ e +$ " " $+ map(v') +$ ".". Finally, the function sparql : $T_C \mapsto SQ$ is defined as shown in Function 1 by tree traversal starting from the root of T and stopping when a non-variable node has been reached.

```
1 $query =$ "SELECT ?x0 { ?x0 ?y ?z . ";
2 $nodequeue = [root(T)]$;
3 while $nodequeue$ $not$ $empty$ do
4     $v = \text{poll}(nodequeue)$ // pick and remove first node ;
5     foreach $edge$ $(v, e, v')$ $in$ $E(T)$ do
6         $query$ $+=$ $mapedge((v, e, v'))$ ;
7         if $\ell(v') =?$ then add $v'$ at the end of $nodequeue$
8 $query$ $+=$ "}";
9 return $query$
```

Function 1. `sparql(`T`)`

2.2 Mapping Resources to Query Trees

Each resource in an RDF graph can be mapped to a query tree. Intuitively, the tree corresponds to the neighbourhood of the resource in the graph. In order to map a resource to a tree, we have to limit ourselves to a recursion depth for reasons of efficiency. This recursion depth corresponds to the maximum nesting of triple patterns, which can be learned by the QTL algorithm, which we will detail in Section 4. Another way to view a query tree for a resource is that it defines a very specific query, which contains the resource itself as result (if it occurs at least once in the subject position of a triple in the knowledge base). The formal definition of the query tree mapping is as follows:

Definition 4 (Resource to Query Tree Mapping). *A resource r in an RDF graph $G = (V, E, \ell)$ is mapped to a tree $T' = (V', E', \ell')$ with respect to a recursion depth $d \in \mathbb{N}$ as follows: $V' = \{v_p \mid v_p \in V,$ there is a path p of length $l \leq d$ from r to v in $G\}$, $E' = V' \times R \times V' \cap E$, $\ell' = \ell_{|V'}$. The result of the function $map : R \times G \times \mathbb{N} \rightarrow \mathcal{T_C}$ is then defined as $T := T'[root(T') \mapsto ?]$.*

Query trees act as a bridge between the description of a resource in an RDF graph and SPARQL queries containing the resource in their result set. Using them enables us to define a very efficient learning algorithm for SPARQL queries. Note that a query tree T does not contain cycles, whereas an RDF graph G can, of course, contain cycles. Also note that query trees intentionally only support a limited subset of SPARQL.

3 Operations on Query Trees

In this section, we define operations on query trees, which are the basis of the QTL algorithm. We define a subsumption ordering over trees, which allows to apply techniques from the area of Inductive Logic Programming [13]. Specifically, we adapt least general generalisation and negative based reduction [2].

3.1 Query Tree Subsumption

In the following, we define query tree subsumption. Intuitively, if a query tree T_1 is subsumed by T_2, then the SPARQL query corresponding to T_1 returns fewer results than the SPARQL query corresponding to T_2. The definition of query tree subsumption will be done in terms of the SPARQL algebra. Similar as in [1,14], we use the notion $[[q]]_G$ as the evaluation of a SPARQL query q in an RDF graph G.

Definition 5 (Query Tree Subsumption). *Let T_1 and T_2 be complete query trees. T_1 is subsumed by T_2, denoted as $T_1 \preceq T_2$, if we have $[[sparql(T_1)]]_G(?x0) \subseteq [[sparql(T_2)]]_G(?x0)$ for any RDF graph G.*

Definition 6 ($\leq$ relation). *For query trees T_1 and T_2, we have $T_1 \leq T_2$ iff the following holds:*

1. *if $\ell(root(T_2)) \neq ?$, then $\ell(root(T_1)) = \ell(root(T_2))$*
2. *for each edge $(root(T_2), p, v_2)$ in T_2 there exists an edge $(root(T_1), p, v_1)$ in T_1 such that:*
   (a) *if $\ell(v_2) \neq ?$, then $\ell(v_1) = \ell(v_2)$*
   (b) *if $\ell(v_2) = ?$, then $T(v_1) \leq T(v_2)$ (see Definition 2)*

We define $T_1 \simeq T_2$ as $T_1 \leq T_2$ and $T_2 \leq T_1$. $T_1 < T_2$ is defined as $T_1 \leq T_2$ and $T_1 \not\simeq T_2$.

The following is a consequence of the definition of $\leq$. It connects the structure of query trees with the semantics of SPARQL.

Proposition 1. *Let T_1 and T_2 be complete query trees. $T_1 \leq T_2$ implies $T_1 \preceq T_2$.*

Proof. We prove the proposition by induction over the depth of T_2. Let G be an RDF graph.

Induction Base ($depth(T_2) = 0$): In this case, $[[sparql(T_2)]]_G(?x0)$ is the set of all resources occurring in subjects of triples in G and, therefore, $T_1 \preceq T_2$.

Induction Step ($depth(T_2) > 0$): $sparql(T_1)$ and $sparql(T_2)$ have a basic graph pattern in their WHERE clause, i.e. a s set of triple patterns. Due to the definition of $sparql$, the triple patterns with subject $?x0$ have one of the following forms: 1.) $?x0\ ?y\ ?z$ 2.) $?x0\ p\ m$ with $m \in R \cup L$ 3.) $?x0\ p\ ?xi$. Each such pattern in a SPARQL query is a restriction on $?x0$. To prove the proposition, we show that for each such triple pattern in $sparql(T_2)$, there is a triple pattern in $sparql(T_1)$, which is a stronger restriction of $?x0$, i.e. leads to fewer results for $?x0$. We do this by case distinction:

1. $?x0\ ?y\ ?z$: The same pattern exists in $sparql(T_1)$.
2. $?x0\ p\ m$ with $m \in R \cup L$: Thus, there is an edge $(root(T_2), p, v_2)$ with $\ell(v_2) = m$ in T_2. Because of the definition of $\leq$, there is an edge $(root(T_1), p, v_1)$ with $\ell(v_1) = m$ in T_1, which leads to the same triple pattern in $sparql(T_1)$.
3. $?x0\ p\ ?xi$: This means that there is an edge $(root(T_2), p, v_2)$ with $\ell(v_2) = ?$ in T_2. Let $(root(T_1), p, v_1)$ with $\ell(v_1) = s$ be a corresponding edge in T_1 according to the definition of $\leq$. We distinguish two cases: a) $s \neq ?$. In this case, $sparql(T_1)$ contains the pattern $?x0\ p\ s$, which is a stronger restriction on $?x0$ than $?x0\ p\ ?xi$. b) $s = ?$. In this case, $sparql(T_1)$ contains the pattern $?x0\ p\ ?xj$. By induction, we know $T(v_1) \leq T(v_2)$ and, consequently, $[[sparql(T(v_1))]]_G(?x0) \subseteq [[sparql(T(v_2))]]_G(?x0)$, i.e. the pattern is a stronger restriction on $?x0$, because there are fewer or equally many matches for $?xj$ than for $?xi$. □

The proposition means that whenever $T_1 \leq T_2$, the result of the SPARQL query corresponding to T_1 does not return additional results compared to the SPARQL query corresponding to T_2. Note that the inverse of the proposition does not hold: If a query q_1 returns fewer results than a query q_2, this does not mean that $T_1 \leq T_2$ for the corresponding query trees, because q_1 and q_2 can be structurally completely different queries.

3.2 Least General Generalisation

The least general generalisation (lgg) operation takes two query trees as input and returns the most specific query tree, which subsumes both input trees. We first define the operation *lgg* algorithmically and then proof its properties.

```
init $T = (V, E, \ell)$ with $V = \{v\}$, $E = \emptyset$, $\ell(v) = ?$;
if $\ell(v_1) = \ell(v_2)$ then $\ell(v) = \ell(v_1)$;
foreach $p$ in $\{p' \mid \exists v_1'.(v_1, p', v_1') \in E(T_1)$ and $\exists v_2'.(v_2, p', v_2') \in E(T_2)\}$ do
    foreach $v_1'$ with $(v_1, p, v_1') \in E(T_1)$ do
        foreach $v_2'$ with $(v_2, p, v_2') \in E(T_2)$ do
            $v' = root(\mathrm{lgg}(T(v_1'), T(v_2')))$; add = true;
            foreach $v_{prev}$ with $(v, p, v_{prev}) \in E(T)$ do
                if $add = true$ then
                    if $T(v_{prev}) \leq T(v')$ then add = false;
                    if $T(v') < T(v_{prev})$ then remove edge $(v, p, v_{prev})$ from $T$;
            if $add = true$ then add edge $(v, p, v')$ to $T$;
return $T$
```

Function 2. `lgg(`T_1, T_2`)`

Function 2 defines the algorithm to compute least general generalisations of query trees. It takes two query trees T_1 and T_2 as input and returns T as their lgg. T is initialised as empty tree in Line 1. The next line compares the labels of the root nodes of T_1 and T_2. If they are equal, then this label is preserved in the generalisation, otherwise ? is used as label. Line 3 groups outgoing edges in the root nodes of T_1 and T_2 by their property label – only if a property is used in both trees, it will be part of the lgg. Line 4 and 5 are used for comparing pairs of edges in T_1 and T_2. For each combination, the lgg is recursively computed. However, in order to keep the resulting tree small, only edges which do not subsume another edge are preserved (Lines 7 to 10). Finally, Line 11 adds the computed lgg to the tree T, which is returned. *lgg* is commutative. We use $lgg(\{T_1, \dots, T_n\})$ as shortcut notation for $lgg(T_1, lgg(T_2, \dots, T_n) \dots)$.

Proposition 2. *Let lgg be defined as in Function 2, T_1 and T_2 be trees and $T = lgg(T_1, T_2)$. Then the following results hold:*

1. *$T_1 \leq T$ and $T_2 \leq T$ (i.e. lgg generalises)*
2. *for any tree T', we have $T_1 \leq T'$, $T_2 \leq T'$ implies $T \leq T'$ (i.e. lgg is least)*

Proof. The proofs are as follows:

1.) We prove by induction over the depth of T. Without loss of generality, we show $T_1 \leq T$.

Induction Base ($depth(T) = 0$): If $\ell(root(T)) \neq ?$, then by Function 2 $\ell(root(T_1)) = \ell(root(T))$ (see also table below).

Induction Step ($depth(T) > 0$): We have to show that for an edge $e = (root(T), p, v) \in E(T)$ the conditions in Definition 6 hold. Due to the definition of Function 2, $e = (root(T), p, root(\text{lgg}(T(v_1), T(v_2))))$ was created from two edges $(root(T_1), p, v_1) \in E(T_1)$ and $(root(T_2), p, v_2) \in E(T_2)$. If $\ell(v) \neq ?$ (Definition 6, condition 1), then $\ell(v_1) = \ell(v)$ by Line 2 of Function 2. If $\ell(v) = ?$ (condition 2), then $T(v_1) \leq T(v)$ follows by induction.

2.) We use induction over the depth of T.

Induction Base ($depth(T) = 0$): We first show $depth(T') = 0$. By contradiction, assume that T' has at least one edge. Let p be the label of an outgoing edge from the root of T'. By Definition 6, both T_1 and T_2 must therefore also have outgoing edges from their respective root nodes with label p. Consequently, $T = lgg(T_1, T_2)$ has an outgoing edge from its root by Function 2 (Lines 4-11 create at least one such edge). This contradicts $depth(T) = 0$.

We make a complete case distinction on root node labels of T_1 and T_2 (note that $m \neq ?$, $n \neq ?$):

$\ell(root(T_1))$	$\ell(root(T_2))$	$\ell(root(T))$ according to Function 2
m	m	m
m	$n (\neq m)$	?
m	?	?
?	m	?
?	?	?

In Row 1, $\ell(root(T'))$ is either m or ?, but in any case $T \leq T'$. For Rows 2-5, $\ell(root(T')) = ?$, because otherwise $T_1 \not\leq T'$ or $T_2 \not\leq T'$. Again, we have $T \leq T'$.

Induction Step ($depth(T) > 0$): Again, we show $T \leq T'$ using Definition 6:

Condition 1: $? \neq \ell(root(T')) = \ell(root(T_1)) = \ell(root(T_2))$ (from $T_1 \leq T'$ and $T_2 \leq T'$) $= \ell(root(T))$ (from Function 2)

Condition 2: Let $(root(T'), p, v')$ be an edge in T'. Due to $T_1 \leq T'$ and $T_2 \leq T'$, there is an edge $(root(T_1), p, v_1)$ in T_1 and an edge $(root(T_2), p, v_2)$ in T_2.

2a): $? \neq \ell(v') = \ell(v_1)) = \ell(v_2)$ (from $T_1 \leq T'$ and $T_2 \leq T'$) $= \ell(v)$ (from Function 2)

2b): Due to $\ell(v') =?$, we get $T(v_1) \leq T(v')$ and $T(v_2) \leq T(v')$. Hence, we can deduce $T(v) = lgg(T(v_1), T(v_2)) \leq T(v')$ by induction. □

3.3 Negative Based Reduction

Negative based reduction is used to generalise a given tree T using trees $T_1, \ldots, T_n$ as input. For each tree, we assume $T_i \not\leq T$ $(1 \leq i \leq n)$. The idea is to generalise T by removing edges or changing node labels without *overgeneralising*. Overgeneralising means that $T_i \leq T$ for some i $(1 \leq i \leq n)$. Negative based reduction has already been used in ILP and is a relatively simple procedure in most cases. In QTL, it is more involved, which is why we only sketch it here due to lack of space. The basic idea is that upward refinement operations are used on the given query tree T until a negative example is covered. When this happens, QTL queries for results of the SPARQL query corresponding to the refined tree. If there are no new resources compared to the *lgg*, then a different upward refinement path is used. If there are new resources, then a binary search procedure is used to find the most specific tree on the upward refinement path, which still delivers new resources. This tree is then returned by QTL.

4 QTL Algorithm

The Query Tree Learner (QTL) integrates the formal foundations from Sections 2 and 3 into a light-weight learning algorithm. QTL is a supervised algorithm, i.e. it uses positive and negative examples as input. In this case, an example is an RDF resource. In a first step, all examples are mapped to query trees as shown in Algorithm 3. The mapping, specified in Definition 4, requires a recursion depth as input. The recursion depth has influence on the size of the generated tree. It is the maximum depth of the generated query tree and, therefore, also the maximum depth of the learned SPARQL query. The mapping method also requires a method to obtain information about a resource from G. In our implementation, this is done via SPARQL queries with the option to use a cache in order to minimise the load on the endpoint. A properly initialised cache also ensures roughly constant response times for the user.

The next step in QTL is to compute the lgg of all positive examples (Line 2). If this results in a tree, which subsumes negative examples, then no query fitting the positive and negative examples can be learned. This is a consequence of Proposition 2. Usually, this happens when the RDF graph does not contain

```
input  : RDF graph $G$, recursion depth $d$, pos. examples
         $E^+ = \{r_1, \ldots, r_m\} \subset R$, $E^+ \neq \emptyset$, neg. examples $E^- = \{s_1, \ldots, s_n\} \subset R$
output: SPARQL Query $q$
1 $T^+ = \{T_i^+ \mid \exists i.r_i \in E^+, T_i^+ = \text{map}(G, d, r_i)\}$; $T^-$ analogously;
2 $T = T_1^+$;  for $i \leftarrow 2$ to $m$ do $T = \text{lgg}(T, T_i^+)$;
3 if there exists a $T_i^-$ with $T_i^- \leq T$ then print "no learnable query exists" ;
4 if $T^- = \emptyset$ then $T' = \text{pg}(T)$ else $T' = \text{nbr}(T, T^-)$;
5 $q = \text{sparql}(T')$
```

Algorithm 3. QTL Algorithm

necessary features to construct a query. For instance, a user may want to get all cities in France, but the endpoint does not contain properties or classes to infer that some city is located in a particular country.

In Line 4 of the algorithm, negative based reduction (nbr) is used to generalise the lgg. A potentially large tree containing everything the positive examples have in common, is generalised by using negative examples as explained in Section 3. In case there are no negative examples available yet, a different operation, *positive generalisation* (pg) is used. Positive generalisations uses the nbr function, but calls it with a seed of resources which is disjoint with the positive examples. This allows to use QTL as positive only algorithm. Finally, in Line 5 of Algorithm 3, the query tree is converted into a SPARQL query. Some characteristics of QTL in combination with the AutoSPARQL interface are discussed in Section 8.

5 AutoSPARQL User Interface

AutoSPARQL is available at `http://autosparql.dl-learner.org`. It is a rich internet application based on the Google Web Toolkit. AutoSPARQL and the QTL algorithm are part of DL-Learner [9]. Their source code is available in the DL-Learner SVN repository.

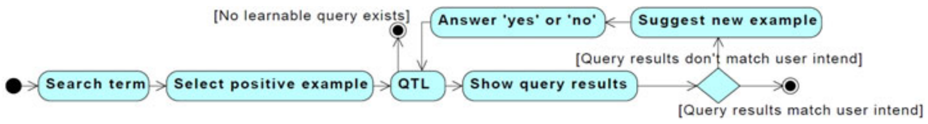
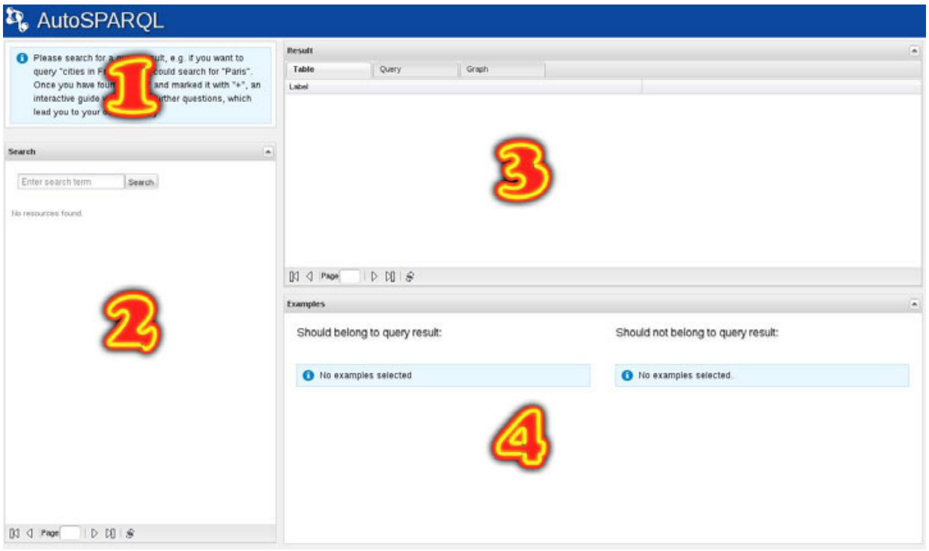

[No learnable query exists]
Answer 'yes' or 'no'
Suggest new example
[Query results don't match user intend]
Search term
Select positive example
QTL
Show query results
[Query results match user intend]

Fig. 2. AutoSPARQL Workflow

The tool implements an active learning method as shown in Figure 2. In a first step, the user performs a query and selects at least one of the search results as positive example, i.e. it should be returned as result of the query he constructs. From this, an initial query is suggested by QTL and the user is asked the question whether a certain resource should be included in the result set. After each question is answered, QTL is invoked again. This process is the active learning part of AutoSPARQL. It is iterated until the desired query is found or no learnable query, matching the examples, exists. The user interface allows several other options such as changing previous decisions, deletion of examples or the selection of several positive examples in one through a tabular interface. The following result shows that AutoSPARQL always returns a correct query or replies that no learnable query exists after a finite number of iterations. The proof, which mainly uses the properties of the lgg function, is omitted, because of lack of space.

Proposition 3. *Let $A = \{r_1, \dots, r_n\}$ be a target set of resources in an RDF graph G, $d \in \mathbb{N}$ and assume that a user/oracle answers questions by AutoSPARQL correctly. If there exists a query tree T with depth $\leq d$ such that $[[sparql(T)]]_G(?x0) = A$, then AutoSPARQL learns a tree T' with $[[sparql(T')]]_G(?x0) = A$, else it reports that no such tree exists.*

In order to improve the efficiency, AutoSPARQL can optionally use the natural language query of the user to filter the query trees. If neither the property nor the label of the target node of an edge in a query tree has a sufficiently high string similarity to a phrase or a WordNet-synonym of a phrase in the natural language query, then it is discarded. Four different string metrics are combined to reduce the probability of filtering relevant edges. If this filter in AutoSPARQL is enabled, the completeness result above no longer holds, because there is non-zero probability that a relevant edge might be filtered.


Please search for a ult, e.g. if you want to query "cities in F could search for "Paris". Once you have foun and marked it with "+", an interactive guide rther questions, which lead you to your

No resources found.

Should belong to query result:

No examples selected

Should not belong to query result:

No examples selected

Fig. 3. Screenshot of initial AutoSPARQL user interface: It consists of four areas (1) question panel (2) search panel (3) query result panel and (4) example overview panel

After QTL has been invoked through a question-answer session, AutoSPARQL allows to further fine-tune the query. For instance, users can select which properties to display, ordering by a property and language settings. As an example, a typical AutoSPARQL session for learning the following query could look as follows. Note that the resources are displayed via a knowledge base specific template.

```
PREFIX dbpedia:  <http://dbpedia.org/resource/>
PREFIX dbo:  <http://dbpedia.org/ontology/>
SELECT ?band ?label ?homepage ?genre WHERE {
?band a dbo:Band .
?band rdfs:label ?label .
OPTIONAL { ?band foaf:homepage ?homepage } .
?band dbo:genre ?genre .
?genre dbo:instrument dbpedia:Electric_guitar .
?genre dbo:stylisticOrigin  dbpedia:Jazz .
}
ORDER BY ?label LIMIT 100
```

- search for "bands with a genre which mixes electric guitars and Jazz"
- resource: `dbpedia:Foals` answer: YES
- resource: `dbpedia:Hot_Chip` answer: NO
- resource: `dbpedia:Metalwood` answer: YES
- resource: `dbpedia:Polvo` answer: YES
- resource: `dbpedia:Ozric_Tentacles` answer: YES
- resource: `dbpedia:New_Young_Pony_Club` answer: NO
- select "genre" as property to return and "homepage" as additional property
- click on "label" column head to order by it and adjust limit

After that, the query can be saved and a URL is provided for the user to call it. Results will be cached with a configurable timeout on the AutoSPARQL server, such that users can efficiently embed it in websites. In particular, in combination with DBpedia Live, this allows users to include up-to-date information in homepages, blogs or forums.

6 Evaluation

We used the benchmark data set of the 1st Workshop on Question Answering over Linked Data (QALD)[9], which defines 50 questions to DBpedia and their answers. From those queries, we filtered those, which return at least 3 resources. This excludes questions asking directly for facts instead of lists. Furthermore, we filtered queries, which are not in the target language of AutoSPARQL, e.g. contain UNION constructs. The resulting evaluation set contains 15 natural language queries. Since answers for all questions were given, we used them as oracle, which answers "YES" when a resource is in the result set and "NO" otherwise. We seeded the positive examples by using the search function of Wikipedia. At

[9] `http://www.sc.cit-ec.uni-bielefeld.de/qald-1`

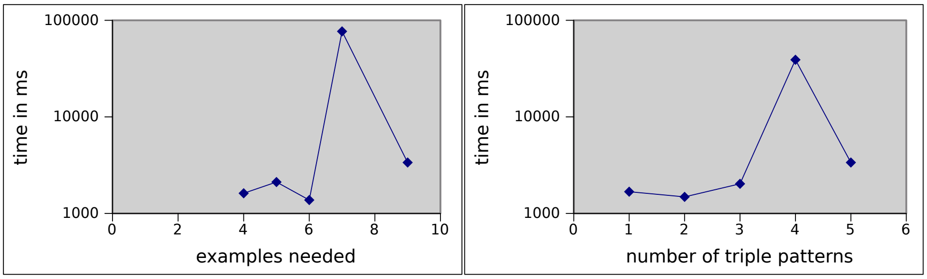


Fig. 4. Statistics showing that roughly 1 s per example are needed (left) and roughly constant time independent of the number of triple patterns (right). The peak in both images is caused by only one single question. All other questions needed less than 10 s to be learned correctly.

most 3 positive examples from the top 20 search results were selected. If less than 3 positive examples are in the top 20, then we picked positive examples from the answer set. In any case, the active learning approach starts with 3 positive and 1 negative examples. The NLP filter, described in the previous section, was switched on since DBpedia has a very large and diverse schema.

The hardware, we used, was a 6-core machine with 2 GB RAM allocated to the VM. We used a recursion depth of 2 for our experiments. We asked questions against a mirror of `http://dbpedia.org/sparql` containing DBpedia 3.5.1. Due to Proposition 3, AutoSPARQL always learns a correct query via this procedure. We were interested in two questions: 1. How many examples are required to learn those queries? 2. Is the performance of AutoSPARQL sufficient?

Regarding the number of examples, we found that 4 (in this case the lgg was already the solution) to 9 were needed and 5 on average. We consider this number to be very low and believe that this is due to the combination of active learning and lggs, which are both known not to require many examples. Most of the examples were positives, which indicates that the questions by AutoSPARQL are mostly close to the intuition of the user, i.e. he is not required to look at a high number of seemingly unrelated RDF resources.

Regarding performance, we discovered that AutoSPARQL requires 7 seconds on average to learn a query with a maximum of 77 seconds. From this, < 1 % of the time are required to calculate the lgg, 73 % to calculate the nbr and 26 % for SPARQL queries to a remote endpoint.

Overall, we consider the performance of AutoSPARQL to be good and a lot of engineering effort was spend to achieve this. The low computational effort required allows to keep response times for users at a minimum and learn several queries in parallel over several endpoints on average hardware.

Apart from the total numbers, we looked at some aspects in more detail. First, we analysed the relation between the number of examples needed and the total time required by AutoSPARQL. Figure 4 shows that roughly the same time per example is needed independent of the total number of examples. This means that the response time for the user after each question remains roughly constant. We also analysed whether there is a relation between the complexity of a query, which we measure in number of triple patterns here, and the time required to learn it. Figure 4 shows that there is no such correlation, i.e. more complex queries are not harder to learn than simple ones. This is common for lgg based approaches. The peak in both diagrams is based on one single question, which was the only question where 7 examples were needed and 1 of 2 questions with 4 triple patterns in the learned query. We discovered that in this case the query tree after the lgg was still very large, so the nbr needed more time than in the other questions.

7 Related Work

In Section 1, we already compared the AutoSPARQL user interface to other techniques like facet-based browsing, visual query builders and interfaces adapted to a specific knowledge base. In this section, we focus on the technical aspects of our solution. AutoSPARQL was mainly inspired by two main research areas: Inductive Logic Programming (ILP) and Active Learning.

The target of ILP [13] is to learn a hypothesis from examples and background knowledge. It was most widely applied for learning horn clauses, but also in the Semantic Web context based on OWL and description logics [6,11,10,7,5] with predecessors in the early 90s [8]. Those approaches use various techniques like inverse resolution, inverse entailment and commonly refinement operators. Least general generalisation, as we used here, is one of those techniques. It has favourable properties in the context of AutoSPARQL, because it is very suitable for learning from a low number of examples. This is mainly due to the fact, that lgg allows to make large leaps through the search space in contrast to gradual refinement. More generally, generate-and-test procedures are often less efficient then test-incorporation as pointed out in an article about the ProGolem system [12], which has influenced the design of our system. ProGolem, which is based on horn logics, also employs negative based reduction, although in a simpler form than in QTL. Drawbacks of lggs usually arise when expressive target languages are used and the input data is very noisy. The latter is usually not a problem in AutoSPARQL, because examples are manually confirmed and can be revised during the learning process. As for the expressiveness of the target language, we carefully selected a fragment of SPARQL, where lggs exist and can be efficiently computed.

Active learning (survey in [15]) aims to achieve high accuracy with few training examples by deciding which data is used for learning. As in AutoSPARQL, this is usually done by asking a human questions, e.g. to classify an example as positive or negative. Active learning has been combined with ILP in [2] to discover gene functions. In our context, an advantage of active learning is that it reduces the amount of background knowledge required for learning a SPARQL query by only considering the RDF neighbourhood of few resources. This way, the burden on SPARQL endpoints is kept as low as possible and the memory requirements for AutoSPARQL are small, which allows to serve many users in parallel. In addition, we use a cache solution, not described here for brevity, to reduce network traffic and allow predictable execution times.

Also related are natural language query interfaces like Google Squared[10], which is easy to use, but less accurate and controllable than AutoSPARQL. In [3], intensional answers have been learned by applying lggs on answers retrieved via the ORAKEL natural language interface. In contrast to our approach, this is done via clausal logic. An integration of natural language interfaces and AutoSPARQL is an interesting target for future work.

8 Discussion and Conclusions

In the final section, we discuss some key aspects of AutoSPARQL/QTL and give concluding remarks.

Efficiency: As demonstrated, one of the key benefits of our approach is its efficiency. This was possible by focusing on a subset of SPARQL and using query trees as a lightweight data structure acting as bridge between the structure of the background RDF graph and SPARQL queries.

Expressiveness: AutoSPARQL supports a subset of SPARQL, which we deem relevant to cover relevant for typical queries by users. However, it certainly does not render SPARQL experts unnecessary, because e.g. when developing Semantic Web applications, more complex queries are needed. Some constructs in SPARQL, e.g. UNION, were avoided, because they would significantly increase the search space and render the approach less efficient. Some extensions of the current expressiveness are already planned and preliminary algorithms drafted, e.g. for learning graph patterns with the same variable occurring more than once in the object of triple patterns and support for different FILTERs.

Low number of questions: Because of Proposition 3, AutoSPARQL is guaranteed to terminate and correctly learn a query tree if it exists. The evaluation shows that a low number of questions is needed to learn typical queries. In the future, we will integrate a natural language interface in AutoSPARQL, such that the first search by the user (see workflow in Figure 2) returns more positive examples, which further simplifies query creation.

Noise: AutoSPARQL/QTL do not support handling noisy data, i.e. it is assumed that the answers to the questions posed by AutoSPARQL are correct. While it is extensible in this direction, we currently pursue the approach of notifying a user when there is a conflict in his choice of positive and negative examples. This can then be corrected by the user. Given the low number of examples in our active learning strategy, this appears to be feasible. However, we envision adding noise handling to QTL for other usage scenarios which do not have these favourable characteristics.

[10] http://www.google.com/squared

Reasoning: Since AutoSPARQL uses triple stores, it depends on the inferences capabilities (if any) of those stores. It is noteworthy that the SPARQL 1.1 working draft contains various entailment regimes[11]. The standardisation of inference in SPARQL is likely to increase support for it in triple stores and, therefore, allow more powerful queries in general and for AutoSPARQL in particular.

Overall: We introduced the QTL algorithm, which is the first algorithm to induce SPARQL queries to the best of our knowledge. The AutoSPARQL interface provides an active learning environment on top of QTL. As we argued in the introduction, the key impact of AutoSPARQL is to provide a new alternative for querying knowledge bases, which is complementary to existing techniques like facet based browsing or visual query builders. We believe that AutoSPARQL is one of the first interfaces to flexibly let non-experts ask and refine non-trivial queries against an RDF knowledge base.

References

1. Angles, R., Gutierrez, C.: The expressive power of SPARQL. In: Sheth, A.P., Staab, S., Dean, M., Paolucci, M., Maynard, D., Finin, T., Thirunarayan, K. (eds.) ISWC 2008. LNCS, vol. 5318, pp. 114–129. Springer, Heidelberg (2008)
2. Bryant, C.H., Muggleton, S., Oliver, S.G., Kell, D.B., Reiser, P.G.K., King, R.D.: Combining inductive logic programming, active learning and robotics to discover the function of genes. Electron. Trans. Artif. Intell. 5(B), 1–36 (2001)
3. Cimiano, P., Rudolph, S., Hartfiel, H.: Computing intensional answers to questions - an inductive logic programming approach. Data Knowl. Eng. 69(3), 261–278 (2010)
4. Clark, L.: Sparql views: A visual sparql query builder for drupal. In: 9th International Semantic Web Conference (ISWC 2010) (Posters&Demo track) (November 2010)
5. Cumby, C.M., Roth, D.: Learning with feature description logics. In: Matwin, S., Sammut, C. (eds.) ILP 2002. LNCS (LNAI), vol. 2583, pp. 32–47. Springer, Heidelberg (2003)
6. Fanizzi, N., d'Amato, C., Esposito, F.: DL-FOIL concept learning in description logics. In: Železný, F., Lavrač, N. (eds.) ILP 2008. LNCS (LNAI), vol. 5194, pp. 107–121. Springer, Heidelberg (2008)
7. Iannone, L., Palmisano, I., Fanizzi, N.: An algorithm based on counterfactuals for concept learning in the semantic web. Applied Intelligence 26(2), 139–159 (2007)
8. Kietz, J.-U., Morik, K.: A polynomial approach to the constructive induction of structural knowledge. Machine Learning 14, 193–217 (1994)
9. Lehmann, J.: DL-Learner: learning concepts in description logics. Journal of Machine Learning Research (JMLR) 10, 2639–2642 (2009)
10. Lehmann, J., Haase, C.: Ideal downward refinement in the $\mathcal{EL}$ description logic. In: De Raedt, L. (ed.) ILP 2009. LNCS, vol. 5989, pp. 73–87. Springer, Heidelberg (2010)
11. Lehmann, J., Hitzler, P.: Concept learning in description logics using refinement operators. Machine Learning Journal 78(1-2), 203–250 (2010)

[11] `http://www.w3.org/TR/sparql11-entailment/`

12. Muggleton, S., Santos, J., Tamaddoni-Nezhad, A.: ProGolem: A system based on relative minimal generalisation. In: De Raedt, L. (ed.) ILP 2009. LNCS, vol. 5989, pp. 131–148. Springer, Heidelberg (2010)
13. Nienhuys-Cheng, S.-H., de Wolf, R. (eds.): Foundations of Inductive Logic Programming. LNCS, vol. 1228. Springer, Heidelberg (1997)
14. Pérez, J., Arenas, M., Gutierrez, C.: Semantics and complexity of SPARQL. 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. 30–43. Springer, Heidelberg (2006)
15. Settles, B.: Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison (2009)

Contextual Ontology Alignment of LOD with an Upper Ontology: A Case Study with Proton

Prateek Jain[1,2], Peter Z. Yeh[2], Kunal Verma[2], Reymonrod G. Vasquez[2], Mariana Damova[3], Pascal Hitzler[1], and Amit P. Sheth[1]

[1] Kno.e.sis Center, Wright State University, Dayton, OH, USA
[2] Accenture Technology Labs, San Jose, CA, USA
[3] Ontotext AD, Sofia 1784, Bulgaria

Abstract. The Linked Open Data (LOD) is a major milestone towards realizing the Semantic Web vision, and can enable applications such as robust Question Answering (QA) systems that can answer queries requiring multiple, disparate information sources. However, realizing these applications requires relationships at both the schema and instance level, but currently the LOD only provides relationships for the latter. To address this limitation, we present a solution for automatically finding schema-level links between two LOD ontologies – in the sense of ontology alignment. Our solution, called BLOOMS+, extends our previous solution (i.e. BLOOMS) in two significant ways. BLOOMS+ 1) uses a more sophisticated metric to determine which classes between two ontologies to align, and 2) considers contextual information to further support (or reject) an alignment. We present a comprehensive evaluation of our solution using schema-level mappings from LOD ontologies to Proton (an upper level ontology) – created manually by human experts for a real world application called FactForge. We show that our solution performed well on this task. We also show that our solution significantly outperformed existing ontology alignment solutions (including our previously published work on BLOOMS) on this same task.

1 Introduction

The Linked Open Data (LOD) is a major milestone towards realizing the Semantic Web vision. A key differentiator of LOD from previous approaches is that data providers are actually creating links across these data sets, which has led to a number of innovative applications spanning multiple, disparate information sources [4]. One missing facet of LOD so far is that these ever-growing ontologies are linked to each other mainly at the instance-level. There are very few schema-level linkages – i.e. links between class hierarchies such as rdfs:subClassOf relations.

A number of researchers [14,13,18][1] (including some of the co-authors of this paper) have argued that without schema-level linkages the LOD cloud will not have semantic-enough information to enable more ambitious, reasoning-based applications of Semantic Web such as Question Answering and Agent-based information brokering. Existing

[1] `http://semtech2010.semanticuniverse.com/sessionPop.cfm?confid=42&proposalid=2854`

© Springer-Verlag Berlin Heidelberg 2011

efforts to develop these types of applications primarily utilize manually created schema-level links between LOD ontologies. For example, *FactForge* enables querying across various LOD ontologies, and utilizes manually developed schema-level mappings of LOD ontologies to an upper level ontology called *Proton* [7].

We believe that manual creation of schema-level mappings across LOD ontologies is not a viable solution given the size of the LOD and the rate at which it is growing. A more automated solution is needed in order for applications such as *FactForge* to effectively scale to (and keep up with) the size of LOD. To this effect, we previously introduced a solution, called Bootstrapping-based Linked Open Data Ontology Matching System (BLOOMS) [13] for automatically finding schema-level links between LOD ontologies. Our previous solution performed well on this task compared to existing solutions such as [12,8,15,16], but there is significant room for improvement.

In this paper, we present a solution called BLOOMS+ which extends our previous solution in two significant ways. BLOOMS+ 1) uses a more sophisticated metric to determine which classes between two ontologies to align, and 2) BLOOMS+ considers contextual information to further support (or reject) an alignment. We present a comprehensive evaluation of BLOOMS+ using schema-level mappings from various LOD ontologies to Proton (an upper level ontology), created manually by human experts. We show that BLOOM+ performed well on this task. We also compare BLOOMS+ to existing ontology alignment solutions (including our previously published work on BLOOMS) on this same task, and show that BLOOMS+ outperformed these solutions. Finally, we present an ablation study, which shows why BLOOMS+ performed well.

The rest of the paper is organized as follows. We first describe the knowledge requirements for BLOOMS+, and why we selected Wikipedia to satisfy these requirements. We then present the BLOOMS+ approach, followed by a comprehensive evaluation of BLOOMS+ and existing solutions. Finally, we present related works along with conclusions and future work.

2 Related Work

To the best of our knowledge, the only other work which exploits contextual information for the purpose of ontology matching has been described in [9]. However, their approach is different from ours as they rely on background knowledge from online ontologies, whereas we rely on a noisy loose categorization of Wikipedia for performing the contextual match. Further, their process relies on identification of contextual relationship using the relationships encoded in the ontologies.

Research in the area of 'Ontology Matching' is very closely related to our body of work. In [10,5] the authors present a survey in the area of ontology matching[2]. The survey work also categorizes the techniques on the basis of external knowledge source utilized by ontology matching systems. While typically, systems utilize a structured source of information such as dictionaries or upper level ontologies, In our previous work in [13] we have presented an approach which exploits a generic and noisy

[2] The ontology matching portal at `http://www.ontologymatching.org/` gives a good review of the state-of-the-art research in this area

categorization system such as Wikipedia in the context of ontology matching. Previously, Wikipedia categorization has been utilized for creating and restructuring taxonomies [20,19].

Another body of related work is identification and creation of links between LOD cloud data sets. In [17] ontology schema matching was used to improve instance co-reference resolution. This helps in cleaning up the data and improving the quality of links at the instance level, but the issue of identifying appropriate relationships at the schema level has not been addressed. The voiD Framework [1] along with the SILK Framework [22] automate the process of link discovery between LOD datasets at the instance level. At the schema level, a notable effort for creating a unified reference point for LOD schemas is UMBEL [3], which is a coherent framework for ontology development which can serve as a reference framework.

3 Knowledge Requirements

BLOOMS+ requires a knowledge source to align two ontologies. The minimum requirements for this knowledge source are:

1. The knowledge source is organized as a class hierarchy where links between classes in this hierarchy capture super and subclass relationships.
2. The knowledge source covers a wide range of concepts and domains, so it can be widely applicable – especially given the wide range of domains covered by the LOD Cloud.

Many knowledge sources – such as WordNet [11], FrameNet [2], SNOMED [6], etc. – satisfy the first requirement, but they fail to satisfy the second. For example, many classes in WordNet and FrameNet are very generic, and hence may have limited utility when aligning domain specific LOD schemas such as Music and Census. SNOMED, on the other hand, captures classes specific to the medical domain, and can be useful for aligning life science LOD schemas. However, it will have limited utility in aligning LOD schemas outside of life science.

BLOOMS+ uses Wikipedia – in particular the category hierarchy in Wikipedia. Although the Wikipedia category hierarchy is not a formal class hierarchy, it still reflects a taxonomy structure. Wikipedia categories roughly correspond to classes in a class hierarchy, and the super and subcategory relationships between these categories roughly correspond to super and subclass relationships. Wikipedia also covers a wide range of categories (over 10 million categories), across many domains. This satisfied the second requirement. Moreover, our previous research [13] has shown that the Wikipedia category hierarchy is effective in aligning LOD schemas.

4 Approach

BLOOMS+ aligns two ontologies through the following steps. BLOOMS+ first uses Wikipedia to construct a set of category hierarchy trees for each class in the source and target ontologies. BLOOMS+ then determines which classes to align by extending BLOOMS in two significant ways. BLOOMS+ 1) uses a more sophisticated measure to compute the similarity between source and target classes based on their category hierarchy trees; and 2) computes the contextual similarity between these classes to further support (or reject) an alignment. Finally, BLOOMS+ aligns classes with high similarity based on the class and contextual similarity.

4.1 Construct BLOOMS+ Forest

BLOOMS+ constructs a set of category hierarchy trees – we call a *BLOOMS+ Forest* F for each class C from the source and target ontologies. For each C, BLOOMS+ tokenizes (and stems) the name of C, and removes stop words from the name.

BLOOMS+ uses the resulting terms as a search string to retrieve relevant Wikipedia pages using Wikipedia search web service.[3] BLOOMS+ treats each page as a possible sense s_i of C and constructs a category hierarchy tree we call a BLOOMS+ tree T_i – for s_i via the following steps.

1. The root of the tree is s_i.
2. The immediate children of s_i are all Wikipedia categories that s_i belongs to.
3. Each subsequent level includes all unique, direct super categories of the categories at the current level.

BLOOMS+ imposes a limit on the depth of the tree being constructed, and defaults this limit to 4. Based on empirical observation depths beyond 4 typically include very general categories (e.g. "Humanities"), which are not useful for alignment. The resulting tree is then added to F.

4.2 Compute Class Similarity

BLOOMS+ compares each class C in the source ontology with each class D in the target ontology to determine their similarity. This is done by comparing each $T_i \in F_C$ with each $T_j \in F_D$ where F_C and F_D are the BLOOMS+ forests for C and D respectively. For each source tree T_i, BLOOMS+ determines its overlap with the target tree T_j.

However, simply counting the number of common nodes the approach used by BLOOMS is insufficient for the following reasons:

– Common nodes that appear deeper in the tree are more generic (and hence less discriminative). They can appear in many BLOOMS+ trees, which can result in false alignments. These nodes should be given less importance when computing the overlap between two trees (and hence the similarity between two classes).
– A large tree can be unfairly penalized because it must have more nodes in common with another tree in order to have a high similarity score. Hence, we need to avoid bias against large trees when computing the overlap.

[3] `http://en.wikipedia.org/w/api.php`

Table 1. Common nodes between the two trees in Figure 1, and their depth. The first column gives the common nodes between the two trees rooted at *Record Label* and *Music Industry*. The second column gives the depth (the distance from root) of these nodes in the BLOOMS+ tree rooted at *Record Label* – i.e. the source tree.

Common Nodes	**Node Depth**
Music_industry	1
Music; Industries; Cultural_economics	2
Industry; Other_special_topics_(economics); Cultural_studies; Economic_systems; Entertainment; Performing_arts; Sound	3

To address these issues, BLOOMS+ uses the following equation to compute the overlap between two BLOOMS+ trees (and hence the similarity of their corresponding classes).

$$Overlap(T_i, T_j) = \frac{log \Sigma_{n \in T_i \cap T_j} (1 + e^{d(n)^{-1} - 1})}{log 2 |T_i|} \quad (1)$$

where $n \in T_i \cap T_j$ are the common nodes between the source and target tree; and $d(n)$ is the depth of a common node n in T_i. The exponentiation of the inverse depth of a common node gives less importance to the node if it is generic, and the log of the tree size avoids bias against large trees. This equation ranges from 0.0 to 1.0 where 0.0 indicates no similarity and 1.0 indicates maximum similarity.

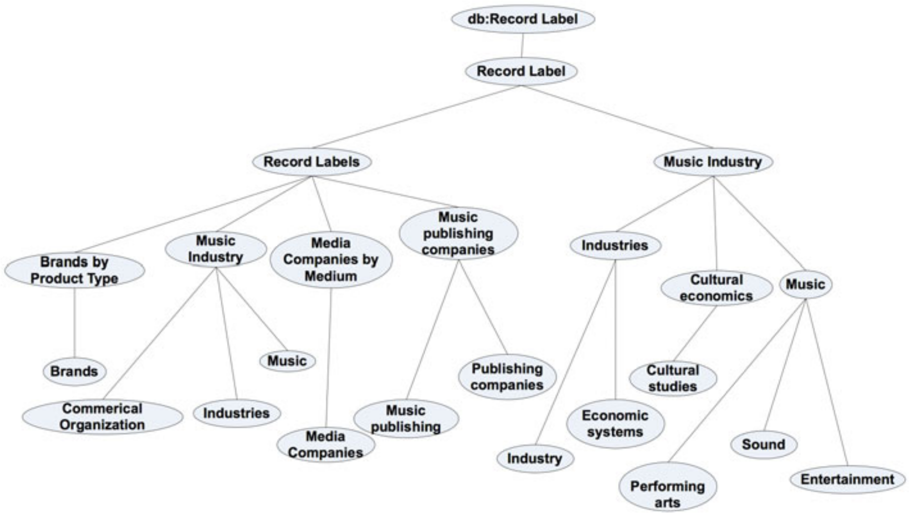
For example, let's assume BLOOMS+ needs to determine whether to align the source class *RecordLabel* from DBpedia with the target class *MusicCompany* from Proton. BLOOMS+ first constructs the BLOOMS+ forests for *RecordLabel* and *MusicCompany*, and Figure 1 shows a BLOOMS+ tree from each forest. BLOOMS+ then identifies the common nodes between these trees, and the depth of these nodes in the tree for the source class (see Table 1).

Finally, the class similarity (see above equation) between *RecordLabel* and *MusicCompany* w.r.t the two BLOOMS+ trees in Figure 1 is 0.79.

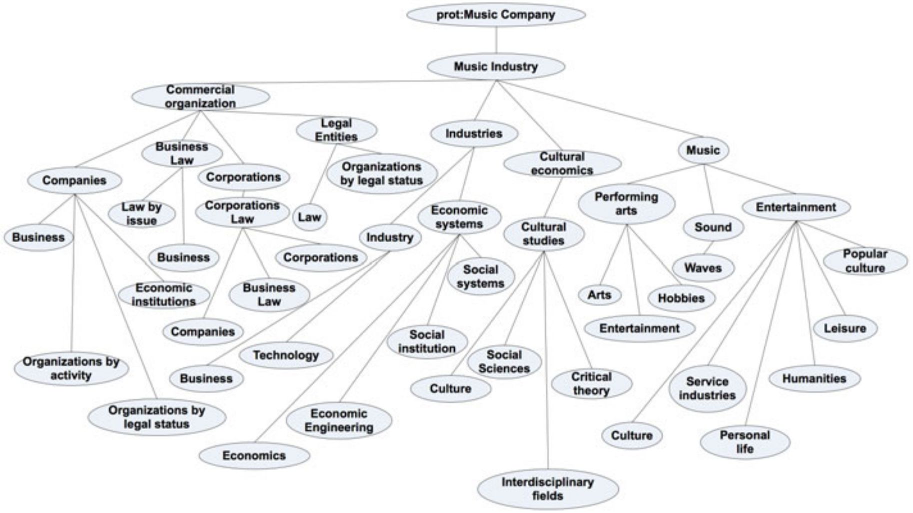
4.3 Compute Contextual Similarity

BLOOMS+ computes the contextual similarity between a source C and target D class to further determine whether these classes should be aligned. A good source of contextual information is the superclasses of C and D from their respective ontologies. If these superclasses agree with each other, then the alignment between C and D is further supported and hence should be given more preference. Otherwise, the alignment should be penalized. For example, the class *Jaguar* might be aligned to the class *Cat*, which seems like a reasonable alignment. However, if *Jaguar* has superclasses such as *Car* and *Vehicle*, and *Cat* has superclasses such as *Feline* and *Mammal*, then the alignment should be penalized because its contextual similarity is low.

BLOOMS+ implements the intuition above in the following way. For each pair wise class comparison (C, D), BLOOMS+ retrieves all superclasses of C and D up to a specified level, which BLOOMS+ defaults to 2. The two sets of superclasses – we'll refer to as $N(C)$ and $N(D)$ – are the neighborhoods of C and D respectively.



(a) BLOOMS+ tree for *RecordLabel* with sense *Record Label*



(b) BLOOMS+ tree for *MusicCompany* with sense *Music Industry*

Fig. 1. BLOOMS+ trees for Record Label 1(a) and Music Company 1(b)

For each BLOOMS+ tree pair (T_i, T_j) between C and D, BLOOMS+ determines the number of superclasses in $N(C)$ and $N(D)$ that are supported by T_i and T_j respectively. A superclass $c \in N(C)$ is supported by T_i if either of the following conditions are satisfied:

- The name of c matches a node in T_i.[4]
- The Wikipedia article (or article category) corresponding to c – based on a Wikipedia search web service call using the name of c – matches a node in T_i.

The same applies for a superclass $d \in N(D)$.

BLOOMS+ computes the overall contextual similarity between C and D with respect to T_i and T_j using the harmonic mean, which is instantiated as:

$$CSim(T_i, T_j) = \frac{2R_C R_D}{R_C + R_D} \tag{2}$$

where R_C (and R_D) are the fraction of superclasses in $N(C)$ (and $N(D)$) supported by T_i (and T_j). We chose the harmonic mean to emphasize superclass neighborhoods that are not well supported (and hence should significantly lower the overall contextual similarity).

Returning to our example, BLOOMS+ needs to compute the contextual similarity for *RecordLabel* and *MusicCompany*. Assuming a level of 2, the neighborhood of *RecordLabel* includes the DBpedia superclasses of *Company* and *Organization*. Both superclasses are supported by the BLOOMS+ tree for *RecordLabel* (see Figure 1(a)), so $R_{RecordLabel}$ is $\frac{2}{2}$. Similarly, the neighborhood of *MusicCompany* includes the Proton superclasses of *CommercialOrganization* and *Organization*. Both superclasses are supported by the BLOOMS+ tree for *MusicCompany* (see Figure 1(b)), so $R_{MusicCompany}$ is also $\frac{2}{2}$. Finally, the overall contextual similarity (see above equation) is 1.0, so BLOOMS+ should give more preference to this alignment.

4.4 Compute Overall Similarity

BLOOMS+ computes the overall similarity between classes C and D w.r.t. BLOOMS+ trees T_i and T_j by taking the weighted average of the class (see Section 4.2) and contextual (see Section 4.3) similarity.

$$O(T_i, T_j) = \frac{\alpha Overlap(T_i, T_j) + \beta CSim(T_i, T_j)}{2} \tag{3}$$

where α and β are weights for the concept and contextual similarity respectively. BLOOMS+ defaults both α and β to 1.0 to give equal importance to each component.

BLOOMS+ then selects the tree pair $(T_i, T_j) \in F_C \times F_D$ with the highest overall similarity score and if this score is greater than the alignment threshold H_A, then BLOOMS+ will establish a link between C and D. The type of link is determined as follows:

- If $O(T_i, T_j) = O(T_j, T_i)$, then BLOOMS+ sets C owl:equivalentClass D.

[4] We define this match as either a direct string match or a substring match.

– If $O(T_i, T_j) < O(T_j, T_i)$, then BLOOMS+ sets C rdfs:subClassOf D.
– Otherwise, BLOOMS+ sets D rdfs:subClassOf C.

Returning to our running example, the overall similarity score between *RecordLabel* and *MusicCompany* is 0.895 (i.e. $\frac{0.79+1.0}{2}$), and BLOOMS+ will establish a link between these classes – assuming the alignment threshold is 0.5. Finally, BLOOMS+ sets *RecordLabel* rdfs:subClassOf *MusicCompany* because $O(T_{\text{Music Industry}}, T_{\text{Record Label}}) > O(T_{\text{Record Label}}, T_{\text{Music Industry}})$.

5 Evaluation

We evaluated the following claims to show that our approach (i.e. BLOOMS+) is effective for ontology alignment over LOD schemas.

Claim 1: BLOOMS+ can outperform state-of-the-art solutions on the task of aligning LOD ontologies.
Claim 2: BLOOMS+ performs well because it accounts for two critical factors when computing the similarity between two classes – 1) the importance of common nodes between the BLOOMS+ trees of the two classes, and 2) bias against large trees.
Claim 3: The performance of BLOOMS+ can be further improved by using contextual information.

5.1 Data Set

We used a real world data set for our evaluation. This data set contains schema-level mappings from three LOD ontologies to Proton, an upper level ontology, with over 300 classes and 100 properties, designed to support applications such as semantic annotation, indexing, and search[21]. The three LOD ontologies include:

– **DBpedia:**[5] The RDF version of Wikipedia, created manually from Wikipedia article infoboxes. DBpedia consists of 259 classes ranging from general classes (e.g. Event) to domain specific ones (e.g. Protein).
– **Freebase:**[6] A large collection of structured data collected from multiple sources such as Wikipedia, Chefmoz, and MusicBrainz. Freebase consists of over 5 million topics and entities, classified into a class hierarchy.
– **Geonames:**[7] A geographic data set with over 6 million locations of interest, which are classified into 11 different classes.

These mappings were systematically created by Knowledge Engineers (KEs) [7] at OntoText for a real world application called *FactForge*[8], which enables SPARQL query over the LOD cloud. The KEs created these mappings, i.e. equivalence and subclass relationships between LOD and Proton classes, based on the definition of the classes and their usage. A total of 544 mappings were created from the three LOD ontologies to Proton (373 for DBpedia, 21 for Geonames, and 150 for Freebase). Table 2 shows examples of these mappings.

[5] `http://downloads.dbpedia.org/3.5.1/dbpedia_3.5.1.owl.bz2`
[6] `http://www.freebase.com/schema`
[7] `http://geonames.org`
[8] `http://factforge.net/`

Table 2. Sample mappings of LOD ontologies to PROTON

Ontology	Class	PROTON Class	Relationship
DBpedia	OlympicResult	Situation	subClassOf
Geonames	Class	LandRegion	subClassOf
Freebase	Event	Event	equivalentClassOf

These mappings provide a good gold standard for our evaluation because:

- The mappings were created by an independent source for a real world use case – unlike existing benchmarks which were created primarily for evaluation purposes. Hence, these mappings reflect the types of relationship that are needed in practice.
- The mappings were created by knowledge engineers through a systematic process [7] and hence are of high quality.
- The mappings cover a diverse set of LOD ontologies. For example, DBpedia and Freebase cover diverse domains such as entertainment, sports, and politics. While Geonames covers only geographic information.

5.2 Experimental Setup

To evaluate Claim 1, we measured the precision and recall of the mappings – from the three LOD ontologies to Proton generated by BLOOMS+. To obtain these measures, we applied BLOOMS+ to each LOD-Proton ontology pair to generate mappings whose overall similarity exceeded an alignment threshold of 0.85 (see Section 4.4). We defined this threshold by systematically analyzing which threshold level produced the best f-measure score. We then compared the resulting mappings for each LOD-Proton ontology pair to their respective gold standard, and said that a mapping between two classes is correct if the gold standard also established a mapping between these two classes using the same relationship i.e. equivalence or subclass. Finally, we defined precision as the number of correct mappings over the total number of mappings generated by BLOOMS+, and recall as the number of correct mappings over all mappings in the gold standard.

We also compared the performance of BLOOMS+ to existing solutions that performed well for LOD ontology alignment, as reported in [13]. These solutions include:

- **BLOOMS:** This is the solution that BLOOMS+ extends [13].
- **S-Match:** This solution utilizes three matching algorithms – basic, minimal, and structure preserving – to establish mappings between the classes of two ontologies [12].
- **AROMA:** This solution utilizes the association rule mining paradigm to discover equivalence and subclass relationships between the classes of two ontologies [8].

To ensure a fair comparison, we used the above methodology to measure precision and recall for each solution, and to define the alignment threshold. The best alignment threshold for BLOOMS is 0.6. The performance of AROMA was not affected by the alignment threshold. It had identical performance for all threshold levels between 0.1 to 1.0. S-Match does not support an alignment threshold. Instead, it returns two sets

Table 3. Results for various solutions on the task of aligning LOD schemas to PROTON. Legend: S-Match-M=Result of S-Match Minimal Set, S-Match-C=Result of S-Match Complete Set, Prec=Precision, Rec=Recall, F=F-Measure PRO=PROTON Ontology, FB=Freebase Ontology, DB=DBpedia Ontology, GEO=Geonames Ontology.

Linked Open Data and Proton Schema Ontology Alignment												
	DB-PRO			GEO-PRO			FB-PRO			Overall		
System	Rec	Prec	F	Rec	Prec	F	Rec	Prec	F	Rec	Prec	F
AROMA	0.19	0.59	0.28	0.04	$\frac{8}{1000}$	0.01	0.31	0.49	0.38	0.22	0.37	0.28
S-Match-M	0.26	0.05	0.08	0.04	$\frac{6}{1000}$	0.01	0.2	0.05	0.08	0.23	0.05	0.08
S-Match-C	0.33	$\frac{3}{1000}$	$\frac{6}{1000}$	0.04	0.009	0.01	0.3	0.4	0.34	0.31	$\frac{4}{1000}$	0.007
BLOOMS	0.48	0.19	0.27	0.04	$\frac{6}{1000}$	0.01	0.28	0.32	0.3	0.42	0.19	0.26
BLOOMS+ No Context	0.77	0.59	0.67	0.04	$\frac{5}{1000}$	0.01	0.48	0.65	0.55	0.66	0.45	0.54
BLOOMS+	0.73	0.90	0.81	0.04	$\frac{5}{1000}$	0.01	0.49	0.59	0.54	0.63	0.55	0.59

Table 4. Sample of **correct** mappings from LOD ontologies to PROTON generated by BLOOMS+

Ontology	**LOD Class**	**PROTON Class**	**Relationship**
DBpedia	RecordLabel	MusicCompany	subClassOf
Geonames	Country	Nation	equivalentClassOf
Freebase	Military_command	Position	subClassOf

of mappings – 1) a minimal set and 2) a complete set, which can be derived from the minimal one. We report both sets in our evaluation.

To evaluate Claims 2 and 3, we created a version of BLOOMS+ without contextual information we call *BLOOMS+ NO-CONTEXT*. The only difference between BLOOMS+ NO-CONTEXT and BLOOMS is the measure used to compute the similarity between two classes (and hence allows us to evaluate Claim 2). The only difference between BLOOMS+ NO-CONTEXT and BLOOMS+ is the use of contextual information (and hence allows us to evaluate Claim 3). We used the above methodology to measure precision and recall for BLOOMS+ NO-CONTEXT, and we set the alignment threshold to 0.85. The evaluation components related to this work are available for download on BLOOMS+ project page.[9]

5.3 Results and Discussion

Table 3 shows the results for all solutions evaluated. Table 4 and Table 5 show examples of correct and incorrect mappings respectively generated by BLOOMS+ from the three LOD ontologies to Proton.

BLOOMS+ performed significantly better than all other solutions in our evaluation on both precision and recall for two LOD-Proton ontology pairs ($p < 0.01$ for χ^2 test in all cases). BLOOMS+ performed well because it utilizes 1) a rich knowledge source – i.e. Wikipedia – to determine the similarity between the classes of two ontologies and 2)

[9] `http://wiki.knoesis.org/index.php/CBLOOMS`

Table 5. Sample of **incorrect** mappings from LOD ontologies to PROTON generated by BLOOMS+

Ontology	LOD Class	PROTON Class	Relationship
DBpedia	Writer	Message	subClassOf
Geonames	Feature	Art	subClassOf
Freebase	Military_command	Event	subClassOf

contextual information from both Wikipedia and the ontologies being aligned. Hence, these results support our first claim that BLOOMS+ can outperform the state-of-the-art on the task of aligning LOD ontologies.

Interestingly, no solution performed well on aligning Geonames with Proton. The only mapping found by BLOOMS+ (and the other solutions) is the class *Country* in Geonames is equivalent to the class *Nation* in Proton. The key reasons for the poor performance include: 1) Geonames has a small number of classes (and hence very limited contextual information) and 2) the names of the classes in Geonames are often vague and ambiguous (e.g. *Code* and *Feature*), which made it difficult to compute their similarity.

BLOOMS+-NO-CONTEXT performed significantly better than BLOOMS w.r.t the overall precision and recall ($p < 0.01$ for χ^2 test on both precision and recall). We attribute this improvement to the only difference between the two solutions. BLOOMS+-NO-CONTEXT uses a more sophisticated measure to compute the similarity between two classes. This measure considers the importance of common nodes between the BLOOMS+ trees of two classes, and avoids bias against large trees. This result supports our second claim that BLOOMS+ performs well because it considers the importance of common nodes and avoids bias against large trees when computing the similarity between two classes.

BLOOMS+ performed significantly better than BLOOMS+-NO-CONTEXT w.r.t to the overall precision ($p < 0.01$ for χ^2 test). Although BLOOMS+ had lower overall recall, this difference was not statistically significant according to the χ^2 test. Moreover, BLOOMS+ had a higher overall f-measure score. We attribute this result to the only difference between these two solutions. BLOOMS+ uses contextual information, and BLOOMS+-NO-CONTEXT does not. Hence, this result supports our third claim that the use of contextual information can further improve performance – in particular precision and f-measure.

6 Conclusion and Future Work

We presented a solution – called BLOOMS+ – for performing ontology alignment. We evaluated BLOOMS+ using schema-level mappings from three LOD ontologies to Proton – created manually by human experts for a real world application called *FactForge* – and showed that BLOOMS+ performed well on this task. We also applied state-of-the-art ontology alignment solutions (including our previously published work on BLOOMS) to this same task, and showed that BLOOMS+ significantly outperformed these solutions on both precision and recall. We also showed that our solution performed well because:

- BLOOMS+ uses a rich knowledge source – i.e. Wikipedia – to determine the similarity between the classes of two ontologies;
- BLOOMS+ accounts for two critical factors when computing he similarity between two classes – 1) the importance of common nodes between the BLOOMS+ trees of the two classes, and 2) bias against large trees.
- BLOOMS+ uses contextual information from both Wikipedia and the ontologies being aligned to further support (or reject) an alignment.

To the best of our knowledge, BLOOMS+ is the only system which utilizes the contextual information present in the ontology and Wikipedia category hierarchy for the purpose of ontology matching.

We plan to utilize BLOOMS+ for the purpose of LOD querying – as outlined in [14] – which requires significant tool support for LOD schema matching in order to scale and keep up with the growth of the LOD cloud. With BLOOMS+, we have made a important step towards solving this bottleneck, and we hope to tackle the problem of querying of LOD cloud next. Finally, we are investigating additional techniques to further improve BLOOMS+ such as incorporating additional contextual information and utilizing other knowledge sources in addition to Wikipedia.

Acknowledgement. The first author was visiting Accenture Technology Labs as a research intern when this work was performed. This work is also funded from NSF Award:IIS-0842129, titled "III-SGER: Spatio-Temporal-Thematic Queries of Semantic Web Data: a Study of Expressivity and Efficiency". Pascal Hitzler acknowledges support by NSF Award:IIS-1017225, titled "TROn - Tractable Reasoning with Ontologies". Mariana Damova acknowledges support by project "RENDER", FP7-ICT-2009-5, Contract no.: 257790[10].

References

1. Alexander, K., Cyganiak, R., Hausenblas, M., Zhao, J.: Describing Linked Datasets – On the Design and Usage of voiD, the 'Vocabulary of Interlinked Datasets'. In: WWW 2009 Workshop on Linked Data on the Web (LDOW 2009), Madrid, Spain (2009)
2. Baker, C., Fillmore, C., Lowe, J.: The berkeley framenet project. In: Proceedings of the 17th International Conference on Computational Linguistics, vol. 1, pp. 86–90. Association for Computational Linguistics, Morgan Kaufmann Publishers (1998)
3. Bergman, M.K., Giasson, F.: UMBEL ontology, volume 1, technical documentation. Technical Report 1, Structured Dynamics (2008), `http://umbel.org/doc/UMBELOntology_vA1.pdf`
4. 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)
5. Choi, N., Song, I.Y., Han, H.: A survey on ontology mapping. SIGMOD Rec. 35(3), 34–41 (2006)
6. Cornet, R., de Keizer, N.: Forty years of SNOMED: a literature review. BMC medical informatics and decision making 8(suppl. 1) (2008), `http://www.biomedcentral.com/1472-6947/8/S1/S2`

[10] `www.render-project.eu`

7. Damova, M., Kiryakov, A., Simov, K., Petrov, S.: Mapping the Central LOD Ontologies to PROTON Upper-Level Ontology. In: Shvaiko, P., Euzenat, J., Giunchiglia, F., Stuckenschmidt, H., Mao, M., Cruz, I. (eds.) Proceedings of the Fifth International Workshop on Ontology Matching. CEUR Workshop Proceedings (November 2010)
8. David, J., Guillet, F., Briand, H.: Matching directories and OWL ontologies with AROMA. In: CIKM 2006: Proceedings of the 15th ACM International Conference on Information and Knowledge Management, pp. 830–831. ACM, New York (2006)
9. Euzenat, J., d'Aquin, M., Sabou, M., Zimmer, A.: Matching ontologies for context. Technical Report NEON/2006/D3.3.1/0.8, INRIA (2007)
10. Euzenat, J., Shvaiko, P.: Ontology matching. Springer, Heidelberg (2007)
11. Fellbaum, C. (ed.): WordNet: An Electronic Lexical Database (Language, Speech, and Communication). illustrated edition edn. The MIT Press, Cambridge (1998), http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20&path=ASIN/026206197X
12. Giunchiglia, F., Autayeu, A., Pane, J.: S-Match: an open source framework for matching lightweight ontologies (2010)
13. 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), http://data.semanticweb.org/conference/iswc/2010/paper/218
14. Jain, P., Hitzler, P., Yeh, P.Z., Verma, K., Sheth, A.P.: Linked Data is Merely More Data. In: Brickley, D., Chaudhri, V.K., Halpin, H., McGuinness, D. (eds.) Linked Data Meets Artificial Intelligence, pp. 82–86. AAAI Press, Menlo Park (2010)
15. Li, J., Tang, J., Li, Y., Luo, Q.: RiMOM: A dynamic multistrategy ontology alignment framework. IEEE Transactions on Knowledge and Data Engineering 21, 1218–1232 (2009)
16. Mascardi, V., Locoro, A., Rosso, P.: Automatic Ontology Matching via Upper Ontologies: A Systematic Evaluation. IEEE Transactions on Knowledge and Data Engineering 22(5), 609–623 (2010)
17. Nikolov, A., Uren, V.S., Motta, E., Roeck, A.N.D.: Overcoming schema heterogeneity between linked semantic repositories to improve coreference resolution. In: Gómez-Pérez, A., Yu, Y., Ding, Y. (eds.) ASWC 2009. LNCS, vol. 5926, pp. 332–346. Springer, Heidelberg (2009)
18. Parundekar, R., Knoblock, C., 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), http://data.semanticweb.org/conference/iswc/2010/paper/334
19. Ponzetto, S.P., Navigli, R.: Large-scale taxonomy mapping for restructuring and integrating wikipedia. In: Boutilier, C. (ed.) Proceedings of the 21st International Joint Conference on Artificial Intelligence, Pasadena, California, USA, July 11-17, pp. 2083–2088 (2009)
20. Ponzetto, S.P., Strube, M.: Deriving a large scale taxonomy from Wikipedia. In: AAAI 2007: Proceedings of the 22nd National Conference on Artificial Intelligence, pp. 1440–1445. AAAI Press, Menlo Park (2007)
21. Terziev, I., Kiryakov, A., Manov, D.: Base upper-level ontology (bulo) Guidance. Technical report, Ontotext Lab, Sirma Group, Deliverable of EU-IST Project IST-2003-506826 (2004), http://proton.semanticweb.org/D1_8_1.pdf
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)

Hide the Stack: Toward Usable Linked Data

Aba-Sah Dadzie[1,⋆], Matthew Rowe[2], and Daniela Petrelli[3]

[1] OAK Group, Dept. of Computer Science, The University of Sheffield, UK
[2] Knowledge Media Institute, The Open University, Milton Keynes, UK
[3] Art & Design Research Centre, Sheffield Hallam University, UK
a.dadzie@dcs.shef.ac.uk, m.c.rowe@open.ac.uk, d.petrelli@shu.ac.uk

Abstract. The explosion in growth of the Web of Linked Data has provided, for the first time, a plethora of information in disparate locations, yet bound together by machine-readable, semantically typed relations. Utilisation of the Web of Data has been, until now, restricted to the members of the community, eating their own dogfood, so to speak. To the regular web user browsing Facebook and watching YouTube, this utility is yet to be realised. The primary factor inhibiting uptake is the usability of the Web of Data, where users are required to have prior knowledge of elements from the Semantic Web technology stack. Our solution to this problem is to *hide the stack*, allowing end users to browse the Web of Data, explore the information it contains, discover knowledge, and *use* Linked Data. We propose a template-based visualisation approach where information attributed to a given resource is rendered according to the *rdf:type* of the instance.

Keywords: Linked Data; Knowledge Visualisation; Information Visualisation; Usable Interfaces; Human-Computer Interaction.

1 Introduction

The Web of Linked Data now connects a wide range of previously disparate and isolated information sources, allowing complex, bespoke queries to be answered that were previously not possible or hard to derive answers for. To tech-savvy users, and in particular, researchers in the Linked Data (LD) community, consumption of LD is easy given their know-how writing SPARQL[1] queries or by applying a *follow-your-nose* principle to sniff out facts and connections between pieces of information. However, to the mainstream web user – who we define as the frequent user who browses web sites, chats with friends on, e.g., Facebook, but has no real knowledge of the intrinsic functionality of the Web they base their interaction on – there exists a gap between exploiting the Web of Data (WoD) to answer queries and the technological know-how to do so. The regular web user does not (and should not need to) know SPARQL, nor RDF[2] (Resource Description Framework), what an ontology or Linked Data is, nor any other element which the Semantic Web (SW) encompasses. This is a problem.

⋆ To whom correspondence should be addressed.
[1] SPARQL query language for RDF: http://www.w3.org/TR/rdf-sparql-query
[2] Resource Description Framework (RDF): http://www.w3.org/RDF

© Springer-Verlag Berlin Heidelberg 2011

To reduce this gap, we propose to make LD usable, allowing end users to embrace the power of the WoD and browse and discover connections between pieces of information and facts as they would on the World Wide Web (WWW – the Readable Web). In bridging the gap we will put a powerful database at the disposal of end users, one which is community maintained and provides answers to unique questions. In essence we propose to *hide the stack* from end users, allowing them to use RDF, SPARQL, ontologies, and all those other elements that make up the *layer cake* [4], without being aware they are doing so. We define this thesis as the *invisible stack* that is expressed by the research question:

How can we make Linked Data usable to real, end users?

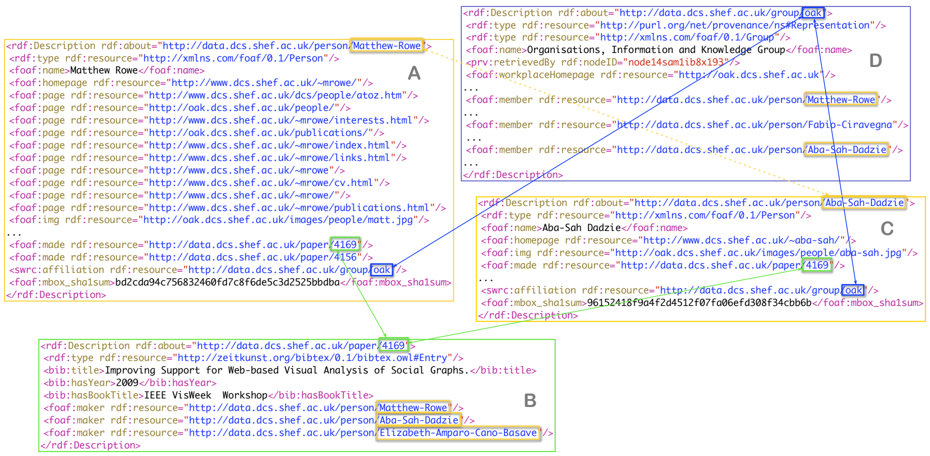
From exploring this question we propose a template-based approach to visualising linked data, that, starting from the underlying data structure associated with a given resource, presents information in a legible and coherent form. The *rdf:type* of a resource provides the primary indicator as to which template(s) to employ, by dereferencing the URI, returning the instance description, and tailoring this information into a legible form, that caters to the user's context, i.e., current task and end goal. Our approach reduces the information load on the user – an endemic problem concerning LD visualisation, given the scale of the WoD – supporting easier interpretation and a coherent view of data.

We have structured this paper as follows: section 2 describes the challenges imposed on the visualisation of LD, based on the current state of the WoD. Section 3 presents related work and how such challenges have been addressed thus far. Section 4 contains our central contribution: our approach to visualising linked data via templates. We define scenarios which motivate and provide references for clarity in explaining our approach. The section concludes by discussing a formative evaluation. Section 6 concludes the paper with the findings and lessons drawn from our work, and plans for future work.

2 Toward Usable Linked Data

Linked Data in its raw form consists of (often very large) sets of RDF statements. RDF was designed to support reading and interpretation of data on the Web by machines. This focus often results in data that is not always easily interpreted by humans, especially outside the SW community. Take, e.g., Fig. 1**B**, which describes a publication in the *Data.dcs*[3] linked dataset; the URI (Uniform Resource Indicator) `http://data.dcs.shef.ac.uk/paper/4169` that references it is fairly cryptic – the only information directly derived from it is that it is a paper belonging to the institution represented by the URI `http://data.dcs.shef.ac.uk` (henceforth abbreviated as '`data.dcs:`'). The complete RDF description, based on its `bibtex`[4] citation, is however easily interpreted by humans; this includes its `bib:title`, publication year (`bib:hasYear`), the book/collection that contains it (`bib:hasBookTitle`), and its authors (`foaf:makers`).

[3] The *Data.dcs*$^\alpha$ LD dataset may be browsed from: `http://data.dcs.shef.ac.uk`
[4] See the BibTeX resource pages at: `http://www.bibtex.org`



```
<rdf:Description rdf:about="http://data.dcs.shef.ac.uk/person/Matthew-Rowe">
 <rdf:type rdf:resource="http://xmlns.com/foaf/0.1/Person"/>
 <foaf:name>Matthew Rowe</foaf:name>
 <foaf:homepage rdf:resource="http://www.dcs.shef.ac.uk/~mrowe/"/>
 <foaf:page rdf:resource="http://www.dcs.shef.ac.uk/dcs/people/atoz.htm"/>
 <foaf:page rdf:resource="http://oak.dcs.shef.ac.uk/people/"/>
 <foaf:page rdf:resource="http://www.dcs.shef.ac.uk/~mrowe/interests.html"/>
 <foaf:page rdf:resource="http://oak.dcs.shef.ac.uk/publications/"/>
 <foaf:page rdf:resource="http://www.dcs.shef.ac.uk/~mrowe/index.html"/>
 <foaf:page rdf:resource="http://www.dcs.shef.ac.uk/~mrowe/links.html"/>
 <foaf:page rdf:resource="http://www.dcs.shef.ac.uk/~mrowe"/>
 <foaf:page rdf:resource="http://www.dcs.shef.ac.uk/~mrowe/cv.html"/>
 <foaf:page rdf:resource="http://www.dcs.shef.ac.uk/~mrowe/"/>
 <foaf:page rdf:resource="http://www.dcs.shef.ac.uk/~mrowe/publications.html"/>
 <foaf:img rdf:resource="http://oak.dcs.shef.ac.uk/images/people/matt.jpg"/>
...
 <foaf:made rdf:resource="http://data.dcs.shef.ac.uk/paper/4169"/>
 <foaf:made rdf:resource="http://data.dcs.shef.ac.uk/paper/4156"/>
 <swrc:affiliation rdf:resource="http://data.dcs.shef.ac.uk/group/oak"/>
 <foaf:mbox_sha1sum>bd2cda94c756832460fd7c8f6de5c3d2525bbdba</foaf:mbox_sha1sum>
</rdf:Description>
```

```
<rdf:Description rdf:about="http://data.dcs.shef.ac.uk/group/oak">
 <rdf:type rdf:resource="http://purl.org/net/provenance/ns#Representation"/>
 <rdf:type rdf:resource="http://xmlns.com/foaf/0.1/Group"/>
 <foaf:name>Organisations, Information and Knowledge Group</foaf:name>
 <prv:retrievedBy rdf:nodeID="node14sam1ib8x193"/>
 <foaf:workplaceHomepage rdf:resource="http://oak.dcs.shef.ac.uk"/>
...
 <foaf:member rdf:resource="http://data.dcs.shef.ac.uk/person/Matthew-Rowe"/>
...
 <foaf:member rdf:resource="http://data.dcs.shef.ac.uk/person/Fabio-Ciravegna"/>
 ...
 <foaf:member rdf:resource="http://data.dcs.shef.ac.uk/person/Aba-Sah-Dadzie"/>
...
</rdf:Description>
```

```
<rdf:Description rdf:about="http://data.dcs.shef.ac.uk/person/Aba-Sah-Dadzie">
 <rdf:type rdf:resource="http://xmlns.com/foaf/0.1/Person"/>
 <foaf:name>Aba-Sah Dadzie</foaf:name>
 <foaf:homepage rdf:resource="http://www.dcs.shef.ac.uk/~aba-sah/"/>
 <foaf:img rdf:resource="http://oak.dcs.shef.ac.uk/images/people/aba-sah.jpg"/>
 <foaf:made rdf:resource="http://data.dcs.shef.ac.uk/paper/4169"/>
...
 <swrc:affiliation rdf:resource="http://data.dcs.shef.ac.uk/group/oak"/>
 <foaf:mbox_sha1sum>96152418f9a4f2d4512f07fa06efd308f34cbb6b</foaf:mbox_sha1sum>
</rdf:Description>
```

```
<rdf:Description rdf:about="http://data.dcs.shef.ac.uk/paper/4169">
 <rdf:type rdf:resource="http://zeitkunst.org/bibtex/0.1/bibtex.owl#Entry"/>
 <bib:title>Improving Support for Web-based Visual Analysis of Social Graphs.</bib:title>
 <bib:hasYear>2009</bib:hasYear>
 <bib:hasBookTitle>IEEE VisWeek  Workshop</bib:hasBookTitle>
 <foaf:maker rdf:resource="http://data.dcs.shef.ac.uk/person/Matthew-Rowe"/>
 <foaf:maker rdf:resource="http://data.dcs.shef.ac.uk/person/Aba-Sah-Dadzie"/>
 <foaf:maker rdf:resource="http://data.dcs.shef.ac.uk/person/Elizabeth-Amparo-Cano-Basave"/>
</rdf:Description>
```

Fig. 1. Extracts from *Data.dcs*, highlighting links between selected resources. **A** and **C** describe two `foaf:Person` resources; **D** their `swrc:affiliation`; **B** a `bib:Entry` resource (a paper), `http://data.dcs.shef.ac.uk/paper/4169`, common to both.

Figs. 1**A** and **C** describe two `foaf:Person` resources (with orange borders): `data.dcs:person/Matthew-Rowe` and `data.dcs:person/Aba-Sah-Dadzie`, respectively, indirectly linked through co-authorship of the paper in **B** (`bib:Entry`, green). **D** describes a second indirect relation, using a broken link – their common `swrc:affiliation`, `data.dcs:group/oak` (blue).

Fig. 1 effectively communicates the inter-relationships because small extracts from *Data.dcs* have been collected in close proximity and specific regions highlighted and colour coded. This, supplemented with bi-directional arrows between resources, results in simple visual encoding that allows the user to gain very quickly an overview and understanding of the relationships within the data. Overlaying the visual encoding on the text allows the user to delve further into the data to retrieve more detail for regions of interest (ROIs), e.g., browsing from the resource `data.dcs:person/Matthew-Rowe` to obtain a human-readable description of the object they *made*: `data.dcs:paper/4169`.

As data set size increases, however, human ability to identify such relationships and retain them in memory decreases significantly. This poses a challenge for the very large amounts of data generated in today's information-rich society, with datasets containing up to millions of entities. *Data.dcs* by comparison is tiny, containing only ∼8000 statements. However even this poses significant cognitive challenges for manual analysis, due to the difficulty obtaining a good mental overview of large amounts of complex, highly interlinked data [8,9,10].

Further difficulties arise when exploring a new environment about which a user has little information, beyond that it contains answers to the questions they wish to ask. While an SW expert would be comfortable with or expect to start browsing LD from a specific URI, the mainstream end user will not have the domain or technical knowledge to do so. In keeping with typical Web usage, a user such as those in our scenarios (see section 4.1) may have obtained a start address from a flier or via a natural language query in a web search engine. From this point they will start to explore the WoD; whether this is the *Readable* or the *Semantic* Web should be transparent to the mainstream user. End users should be given usable tools to explore the WoD, linking to relevant LD or other data in the wider Web, and access to simple methods for exporting the data to alternative readable formats. For the tech-savvy user it is also useful to allow extraction of the underlying RDF data using formal syntax such as SPARQL.

For such interaction to occur, we have identified key usability challenges which currently exist in using LD:

2.1 Challenges to Linked Data Use

Exploration starting point: where to start; existing LD browsers assume the end user will start browsing from a specific, valid URI. How can a visualisation starting point be presented to users in such a way that it is meaningful?

Combating information overload: presenting end users with all the properties of a given resource, along with the relations through which the resource is linked to other entities, leads to information saturation and a dense information space. How can we present this information in a more legible form?

Returning something useful: RDF is the staple recipe for resource descriptions, returning information using this knowledge representation format inhibits comprehension. How can RDF, and the information contained within instance descriptions, be represented in a more legible, manageable form?

Enabling interaction: end users are familiar with the makeup of the Web and its browsable nature. Is it possible to replicate such familiarity which users experience when browsing the WWW on the WoD?

3 Related Work

The SW community has to date largely focused on the use of text-based representations of linked data, often (explicitly or transparently) through the use of SPARQL endpoints. This is due to two main reasons: (1) the infancy of the LD initiative; (2) the focus on prototypes to serve the needs of the SW community and other specialised domains, to generate and analyse LD and related SW data. The most widely referenced LD browsers include *Sig.ma* [20], *Marbles* [3] and *URI Burner*[5]. *Simile*[6] provides access to a set of tools and APIs (Application Programming Interfaces) for presenting and interacting with RDF data.

Haystack [15] was one of the first SW browsers developed, to lower the barrier to consumption of SW resources. It uses stylesheets and views/lenses defined in RDF to aggregate distributed information and customise its presentation for specific users and tasks. Corlosquet et al., [6] describe the extension of *Drupal*[7]

[5] URI Burner: `http://linkeddata.uriburner.com`
[6] Simile: `http://simile.mit.edu`
[7] Drupal Content Management System: `http://drupal.org`

to create enriched web sites, whose models are defined using standard ontologies, in order to embed RDF into the underlying content. Their approach consequently enables the retrieval also of relevant LD on the fly, via SPARQL endpoints.

Textual presentation of RDF data, while familiar and useful to SW expert users, results in high cognitive load for non-technical and non-domain experts, and even for experts, as data amount and interconnectivity increase. Visualisation is acknowledged to enhance knowledge discovery and analytical ability while lowering cognitive load, by harnessing powerful human perception [10,11,17] to enable intuitive construction of an understanding of the structure of large amounts of complex, interacting data. Visualisation affords a number of everyday metaphors, by encoding data attributes into, e.g., graphs, maps and timelines, providing more understandable user interfaces (UIs) over machine-friendly RDF. The LD community is, not surprisingly, examining visual solutions for both technical and mainstream use. These often visualise the RDF graph structure, e.g., [7,9,13,18]; such graphs are very useful to technical experts, whose requirements include inspection to identify errors, retrieve selected data and relations, and develop specialised applications. A well-known RDF visualisation tool is *IsaViz* [13], which overlays RDF graphs with stylesheets based on *Fresnel lenses* [14].

A small number of LD visualisation browsers include *RelFinder* [7], which automates link discovery between user-specified resources. *Tabulator* [5] interprets LD by mapping to standard ontologies such as FOAF[8] (Friend of a Friend), with output to a nested hierarchy and visualisations including map and calendar views. *RDFScape* [18] is a *Cytoscape*[9] plug-in to enhance analysis in Bioinformatics, visualising the results of ontology-based inference with node-link graphs.

However, there is a dearth in applications targeted at mainstream, non-technical use. Hirsch et al., [9] help to fill this gap, by visualising the (semantic) knowledge content in *Freebase*[10] and *Wikipedia*[11] using node-link graphs. They use icons, colour and relative node and edge size to encode data attributes, and draw (labelled) links between clusters of related semantic information. *LESS* [2] supports the creation of (shareable) web-based templates to aggregate and display LD to mainstream users through the familiar presentation methods of the readable Web. *DBPedia Mobile* [3] exploits a geographical metaphor to link and publish information about resources in the user's vicinity. It further lowers the barrier to interaction with LD by enabling the publication of new information about resources in users' physical location to the LD cloud.

Our review of the state of the art, with respect to the challenges outlined in section 2.1, highlights the work to be done to make LD *usable* to non-tech savvy end users. In particular we note the need for solutions which allow more flexible, open-ended knowledge discovery. RelFinder and DBPedia Mobile come closest to fulfilling this, by revealing, using different mechanisms, relations between distinct entities. Additionally we note that low user familiarity with linked

[8] FOAF Ontology Specification: `http://xmlns.com/foaf/spec`
[9] Cytoscape: `http://www.cytoscape.org`
[10] Freebase: `http://www.freebase.com`
[11] Wikipedia: `http://en.wikipedia.org`

datasets – neither the intricacies of the data, nor its entire subject scope – hinders the formulation of initial questions. In such cases a viable means of exploration is required, much as Web browsing functions, to allow intuitive information discovery. In turn, a challenge of such discovery mechanisms is information overload.

Visualisation is essential in providing an entry point into the WoD and combating the cognitive load associated with the use of these large, distributed, highly inter-connected data sets. However visualisation has its own limits. In section 4 we explore the synergies between higher level data overviews and detailed, interactive analysis of ROIs. What constitutes a useful detail view however varies depending on the end user and their task; while we focus on the design of UIs that support mainstream end users we are careful not to ignore the expert user. This paper aims also to address this last challenge – graphical overviews, for instance, allow users to obtain a high level understanding of data structure. The mainstream end user can use these to identify a starting point for their exploration, while the expert will quickly recognise valid patterns and clusters in addition to anomalies in the data structure.

4 Template-Based Visualisation of Linked Data

Differences in users, their tasks and environments mean no one solution can claim to exhaustively meet all requirements for using LD. However, template-based approaches hold significant potential for helping to bridge these challenges. Templates are a useful, flexible tool that may be used to define how to format RDF data into a human-readable representation [2,6,14,15,16], and synthesise related but distributed, heterogeneous information [3,9,18,20], improving management of its knowledge content. In this section we discuss our design rationale for a template-based visualisation approach to presenting linked data, a solution that tackles the challenges identified in section 2.1.

To provide sufficient context to illustrate our work, we detail two scenarios:

4.1 Scenarios of Use

a. *A primary school teacher in Sheffield is preparing a technology project for year 6 pupils that examines the future of the Web. To help her pick a topic that will engage her students she wishes to speak to researchers in the local university exploring leading edge Web technology. She goes to the University of Sheffield web site and navigates to the Department of Computer Science. She finds a link to the research areas...*

b. *Anne has just completed her 'A' levels and received a grant to study Computer Science (CS) at the University of Sheffield. Her parents know that she wishes to work as a lecturer after she graduates. They want to reassure themselves that she will be exposed to a wide range of topics and be able to interact with academics at the forefront of their fields. Therefore they go the University website and navigate to the Department of Computer Science...*

In the remainder of the paper we will complete both stories, illustrating, with *Data.dcs*, how our solution helps these actors retrieve the information they seek.

4.2 Design Rationale and Implementation

Our solution to visualising linked data is to *hide the stack*; by this we refer to utilising SW technologies inherent in LD, but with these elements transparent to the user. In short, this bypasses the complexity associated with RDF and SPARQL by rendering information returned using such standards in a human-legible form. The proposed solution makes use of custom templates loaded based on the *rdf:type* of the resource being viewed by the user. Two views are provided: a *graph view* that draws the relations between resources, displaying information defined using *object* properties; and a co-ordinated *detail view* that presents attributes and information defined using *datatype* properties. Our solution allows the collapsing of detail into *compound nodes*, revealing detail only for the focus. More advanced options are also available to the tech-savvy user, for advanced querying that links directly to the wider WoD using public SPARQL endpoints.

We first describe our design, and how we translated this into the presentation of information using templates. We then summarise the results of a focus group study designed to collect user opinions about the capability enabled for exploratory knowledge discovery and directed search.

The Templates

The first step in our template design is to identify which resource types in a given data set may be of interest to the end user. At the same time a solution is required that does not assume more than cursory knowledge of data structure or content. Well specified LD should, as far as is possible, re-use and/or extend standard ontologies and vocabularies when describing data content. Core concepts from such ontologies and vocabularies are therefore an ideal first point from which to create reusable templates, adaptable to user context and tasks.

We illustrate this for metadata that describes people and information about people, starting from the organisations they work for. Examples of widely applicable ontologies, from which we select relevant classes and properties or relations to define templates, include: **FOAF** to describe people and their relations with other people and organisations; **SWRC** (Semantic Web for Research Communities[12]), relevant to the use cases we discuss. Further, SWRC is easily generalised to other organisational structures; **BibTEX** to describe publications; **PRV,** the Provenance Vocabulary[13], to support verification of data content and quality.

These examples both provide a solution and raise another issue – `Organization` and `Person`, for instance, are defined in both FOAF and SWRC. For the initial versions of the templates we chose to model concepts using the most widely referenced ontology or vocabulary, to increase reusability. So our templates default to, e.g., FOAF for `Organization` and `Person`, but use both `member` from FOAF and `affiliation` from SWRC, to model directed relationships between an `Organization` and a `Person`, respectively. `Documents` (of which `Publications` are a subset) are modelled using BibTEX rather than FOAF or SWRC, with

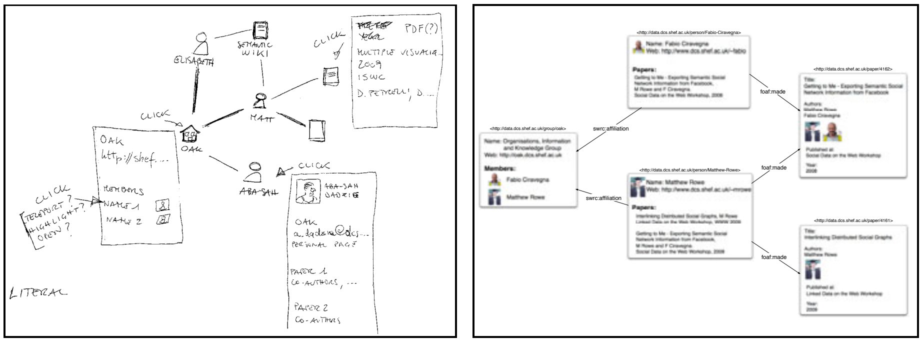
[12] SWRC Ontology Specification: `http://ontoware.org/swrc`

[13] PRV Core Ontology Specification: `http://purl.org/net/provenance/ns#`

the relation between a `Person` and a `bibtex:Entry` modelled using the FOAF property `maker`. In order to apply the templates also to ontologies that redefine commonly referenced concepts we plan to include selectors that allow cascading to cater for redundancy, in addition to equivalent classes across ontologies, and subclasses defined in a single ontology or by extension in a new ontology.

To support extensibility and reusability we define a template for each key resource first using Fresnel lens SPARQL selectors (see top, Fig.5). We then translate these to a set of presentation formats in a Java prototype, using the design ideas expressed in Fig. 2. Three templates that provide customised views over RDF data are illustrated in this paper, using the *Data.dcs* linked dataset, for the resource types:

Group (Organisation): `<http://xmlns.com/foaf/0.1/Group>`
Person: `<http://xmlns.com/foaf/0.1/Person>`
Publication: `<http://zeitkunst.org/bibtex/0.1/bibtex.owl#Entry>`.



(a) (b)

Fig. 2. A design sketch (2a) illustrates the use of icons to encode graph nodes, based on `rdf:type`. Links show primary relationships, e.g. `foaf:Person` *`foaf:made`* `bib:Entry`. Interaction with the graph, e.g. *onNodeClick*, reveals a *group* detail template (left); a *Person* (lower, right); a *Publication* (top, right). 2b illustrates potential implementation as a sub-graph of nodes expanded to show detail, with directed, labelled edges.

Returning to the scenarios, our design aims to support especially the non-expert user (outside the SW and CS domains). This type of user will typically have, at best, a broad idea of where to find the information they seek. They are most likely to exhibit exploratory information seeking behaviour, moving on to directed search once they have a more complete understanding of what knowledge is available. The scenarios have each actor at the start page of the CS department, where they are presented with: traditional, text-based browsing or visualisation-based browsing of the departmental structure. They choose the visualisation over reading the research pages on the web site, because the site explains that the latter displays the relationships between groups and researchers.

From an implementation point of view, that the visualisation is based on *Data.dcs* is (and should be) transparent to the user. We cannot assume that our target end user will have a specific URI to browse from (the root and initial focus of their visual graph). In this case two options are available: (1) presenting the user with a list of potential starting points, e.g., for *Data.dcs*, research groups or their heads – this is however only feasible for a restricted data set, which is seldom the case for LD; (2) random selection, restricted to key resource types. We currently use the second option, rooting the (directed) graph with the first subject read that matches one of the pre-specified resource types of interest – in the *Data.dcs* example this would be a group/affiliation, a person or a publication.

The Graph View

Simply drawing the graph showing the relations within *Data.dcs* results in a representation such as that in Fig. 3a. To reduce the potential for cognitive overload the graph displays only the first two levels from the root at start-up, leaving the user to unfold the graph, up to its leaves. This however quickly results in a dense *web* of nodes, a problem common to node-link graphs [8]. This is seen even with reduced occlusion in Fig. 3a, by abbreviating resource URIs, and with only part of the graph drawn, illustrating expansion from the third to the fourth (out of eleven) levels. Presenting this to users, whether non-expert or expert, only highlights the density and complexity of the information they wish to explore; a more usable solution is necessary.

We resolve this issue first by filtering the dataset, based on key (RDF) resource types, identified through the use of standard templates, similar to the approach in Fresnel [14]: in this case, (research) *group* (`Organisation`), `Person` and *paper* (`Publication`). The ∼8000 statements in *Data.dcs* generate ∼3000 distinct graph nodes; the filters reduce the node count to ∼300. The detail for each instance of these resources is collapsed into what we shall refer to as a *compound* node. The resulting graph, in Fig. 3b, provides a high level overview, with significant reduction in occlusion, that displays key information in the data.

However, this solution presents yet another issue: the filters hide what is useful, detailed information from the user. As discussed also in [8,17] there is the need to support the examination of detail in selected ROIs. We resolve this by maintaining the visual graph as the centre of the user's exploration activity, and couple it with a view that formats for human consumption the detail for each compound node (see Figs. 4 and 5). This allows the user to maintain the context of surrounding information even while exploring ROIs in the co-ordinated detail window (which we describe in the following sub-section).

Colour coding and node size are the two main visual encoding methods for key properties in the graph. *Organisation* nodes and outlinks have a blue border, and *Person* and *Publication*, orange and green, respectively. The focus node has a red fill, and the borders of its immediate neighbours change to red *onMouseHover*. Keyword query matches are filled in pink. Node size may be weighted by the number of outlinks; this is especially useful when child nodes are folded into a parent, or hidden using a filter. In, e.g., Fig 4, group members with relatively

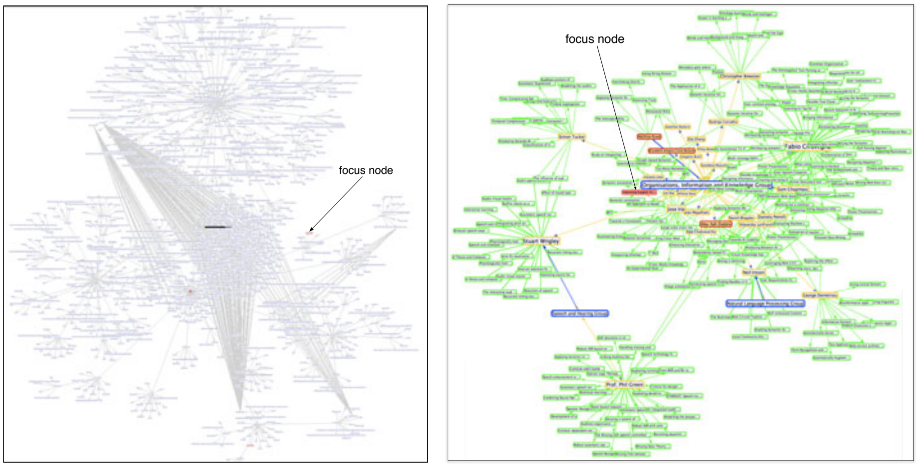
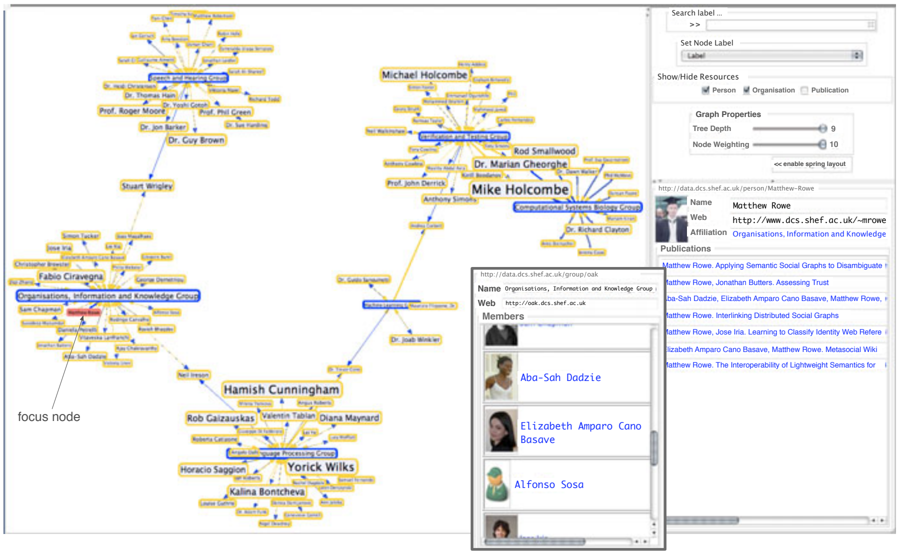


Fig. 3. 3a shows the layout for the first three levels of the complete graph for *Data.dcs*, as it is being expanded to the next level. 3b shows the equivalent graph for the first four levels that collapses detail into compound nodes using a custom filter.

high publication count stand out. The smallest group node has the smallest size (comparing height and node font size; width is determined by label length).

The actors in our scenarios both choose to explore by research group. Anne's parents are simply browsing to get a good feel for the variety and depth of the research carried out. The schoolteacher has a more complex goal, to identify which group is most likely to be able to provide her with information for her technology project. We will concentrate for now on the latter. Moving from Fig. 3b she elects to display only *Organisation* and *Person* nodes (filters – top, right, Fig 4), to remove the clutter of the large number of *Publication*s, as these are currently of secondary interest. She then expands the filtered graph to display all levels, shown in the larger pane in Fig 4. The overall structure of the department is easily discerned. Distinct clusters differentiate each group, while links between them due to researchers in multiple groups span the space between clusters.

A quick scan of group names returns: (1) *Computational Systems Biology* (CompBio), (2) *Machine Learning* (ML), (3) *Natural Language Processing* (NLP), (4) *Organisations, Information and Knowledge* (OAK), (5) *Speech and Hearing* (SpandH) and (6) *Verification and Testing* (VT). The teacher eliminates all but one: OAK. Graph visualisation is able to give a high level overview of such large, interlinked data sets. However, the graphs are not as effective for detailed knowledge exploration. We describe next the rationale behind the template design used to provide human-readable detail, which the teacher will use to delve into the group's research, to determine if it is relevant to her project.



Search label ...
>>
Set Node Label
Label
Show/Hide Resources
[✓ Person] [✓ Organisation] Publication
Graph Properties
Tree Depth 9
Node Weighting 10
<< enable spring layout

http://data.dcs.shef.ac.uk/person/Matthew-Rowe
Name Matthew Rowe
Web http://www.dcs.shef.ac.uk/~mrowe
Affiliation Organisations, Information and Knowledge
Publications
Matthew Rowe. Applying Semantic Social Graphs to Disambiguate
Matthew Rowe, Jonathan Butters. Assessing Trust
ba-Sah Dadzie, Elizabeth Amparo Cano Basave, Matthew Rowe,
Matthew Rowe. Interlinking Distributed Social Graphs
Matthew Rowe, Jose Iria. Learning to Classify Identity Web Refere
Elizabeth Amparo Cano Basave, Matthew Rowe. Metasocial Wiki
Matthew Rowe. The Interoperability of Lightweight Semantics for

http://data.dcs.shef.ac.uk/group/oak
Name Organisations, Information and Knowledge Group
Web http://oak.dcs.shef.ac.uk
Members
Aba-Sah Dadzie
Elizabeth Amparo Cano Basave
Alfonso Sosa

Fig. 4. Filtering out publications highlights links across groups in the departmental structure. The focus node, `http://data.dcs.shef.ac.uk/person/Matthew-Rowe`, is highlighted in red, and its detail shown in the template window on the right. The group template for this `Person` is populated and superimposed on the graph.

The Detail View

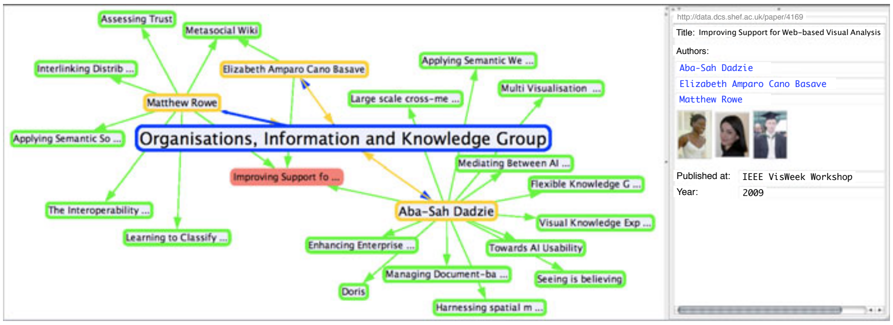
The teacher browses the information about the OAK `Group` by clicking on nodes for `members` (in the graph). This updates the coupled detail template window, allowing her to examine each `Person` resource in more detail. Alternatively, she could select the OAK Research `Group` node, to populate and display the `Organisation` detail template (overlay, Fig. 4), and switch to browse the detail for its `members` via the thumbnails linking to each `Person` template.

The `member` "*Matthew Rowe*" lists a set of publications that may be relevant (right pane, Fig. 4); the teacher selects one, "*Improving Support for Web-based Visual Analysis of Social Graphs*", to examine in greater detail – this is the human-readable label for the URI `data.dcs:paper/4169`. Fig. 5 shows the SPARQL query template (based on the Fresnel template for this resource) used to retrieve the full view for a publication, and the result of applying this template to `data.dcs:paper/4169`. This is the same node highlighted in Fig. 3; the difference in ability to interpret what the node represents can be clearly seen.

The teacher reduces the depth of the graph to two levels, removes the *Publication* filter, and zooms in to the detail for the three co-authors (bottom, Fig. 5). This allows her to scan (in the graph) the titles of other publications written by the authors, while maintaining the focus on the publication of interest in the detail template window (right). Armed with this information she goes to the web pages of each of the co-authors, via the `Person` detail view. She will also browse the `group`'s web pages, to extract more information about their projects.

SPARQL query template for the full publication view for mock URI <*data.dcs:publicationUri*>

```
PREFIX foaf: <...>    PREFIX bib: <...>
SELECT DISTINCT ?publicationTitle ?year ?bookTitle ?personUri ?author ?imageUri
WHERE {
    <data.dcs:publicationUri> bib:title ?publicationTitle ;
        bib:hasYear ?year ;
        bib:hasBookTitle ?bookTitle ;
        foaf:maker ?personUri .
        ?personUri foaf:name ?author ;
        foaf:img ?imageUri
} ORDER BY DESC(?year) ?publicationTitle
```



http://data.dcs.shef.ac.uk/paper/4169

Title: Improving Support for Web-based Visual Analysis

Authors:
Aba-Sah Dadzie
Elizabeth Amparo Cano Basave
Matthew Rowe

Published at: IEEE VisWeek Workshop

Year: 2009

Fig. 5. The detail view for the `Publication` `http://data.dcs.shef.ac.uk/paper/4169` is shown on the right, while the graph in the centre is zoomed in to show other (`Publication`) links from the `Person` resources of interest, its co-authors (or `makers`)

5 Formative Evaluation

To evaluate our approach a focus group study was carried out to: (1) review the requirements for applications for consuming LD; (2) determine if the co-ordinated views allow both mainstream users and technical and domain experts to obtain a good understanding of data content and structure; (3) identify where the design required revision prior to formal, summative, usability evaluation.

As part of a conference tutorial[14] to study the importance of Human-Computer Interaction (HCI) and user-centred interface design when building end user tools on SW technology, 14 participants from the SW community and related research areas gave feedback on the use of the prototype for information exploration and retrieval tasks. The participants, who would be considered to be tech-savvy, assumed the role of expert reviewers (see [12,19]) inspecting the interface from the point of view of the target end user. At this stage in the design we consider, in addition to the mainstream user, the technical user building SW tools. Such users need support also to identify: (1) incomplete data; (2) errors, e.g., redundancy and incorrect linking during automatic LD generation; (3) key properties of the data and optimal ways for presenting these and their inter-relationships.

The participants were given an overview of the prototype and its design rationale, and challenges to LD exploration and use. They then carried out a practice

[14] 'Essential HCI for the Semantic Web' @ ESWC 2010: `http://www.eswc2010.org`

task on a small data set based on a television series, to extract structural information. This was followed by a more complex exploration task, to retrieve information from *Data.dcs* on, among others, collaboration across groups, and researchers with multiple affiliations and relatively high publication count.

In focus groups of up to 4, the participants then reviewed the prototype as part of a participatory design activity. They assessed how well the UI supported user requirements, based on three main criteria: (1) effectiveness, i.e., how successful they were in discovering the information required; (2) efficiency, i.e., time to complete tasks; (3) user satisfaction, guided by a questionnaire.

5.1 Results

Because the evaluation was formative the focus was on collecting qualitative information to validate the requirements that fed into the UI design, and determine how effectively this had been translated to supporting LD consumption. Information was obtained by observing the participants, supplemented by a debrief in which the outcomes of the focus group study were discussed. This phase generated a number of post-it notes that were clustered and analysed.

The participants found, overall, the graphs to be quite expressive and effective in giving a sense of the data distribution. However those not accustomed to graph manipulation had difficulty controlling the display, due first to the large graph size, and also the (spring) layout algorithm that alters the layout to re-centre on focus change. Comments such as: "*eventually you got a big picture of the data* " and "*I liked the direct manipulation but the graph should stay put [when I click]*", show both frustration and appreciation. Another key feature recognised was the effectiveness in displaying detail for key resources: "*in a neat and concise way*".

The prototype was seen to have potential for exploring and debugging LD. The capability for keyword search was much appreciated. This was reinforced by requests for more selective filtering, both naïve and expert (e.g., via SPARQL). Though the latter could be motivated by the participants' technical expertise, it is more likely to indicate the need for tools for both (usable) browsing and search to be integrated into the same interface. Suggestions for improving interaction with the knowledge content included explicit highlighting of search paths in the graph; browsing through previous actions (history); additional options for navigation, e.g., *porting* between disconnected sub-graphs.

6 Conclusions

We have presented a solution to making LD *usable* by mainstream end users. This seeks to *hide the stack* from such users, enabling use of the rich network of information on offer, but without requiring knowledge of elements in the SW technology stack to process the information. For instance, the templates in our approach are, in essence, SPARQL queries, the bound variables within which are tied to regions in the information presentation view. The user need not know that SPARQL is utilised, nor that the resource description is returned as RDF.

We highlighted four key challenges for interacting with LD that we address as follows: the selection of an *exploration starting point* can be achieved via lists of potential starting points, allowing the user to choose the resource to focus on; or by randomly selecting a resource from which to start browsing.

For *combating information overload* we utilise two types of views, based on templates: a *graph* and a *detail* view. The former provides an overview of the network structure surrounding a resource of interest. Our default displays resources up to two steps (levels in the graph) from the focus on the WoD. This provides a clear picture that describes the relations between external resources and possible paths and transitions through the space, while preventing information overload. This is consistent with recent work in [7], where relations are revealed between DBPedia concepts in a graph. Our second view displays the properties of the focus resource, similar to work by [2], by rendering the RDF response from dereferencing the URI into a legible form. By marrying the two views we are able to separate out the logical components of relations and properties, where we use the latter to show datatype properties of the resource, and the former to show the context and role of the resource within the wider WoD.

The use of templates has also addressed the challenge of *returning something useful*, by providing information in a format that end users are able to understand and interpret. Colour coding resources in the graph view based on *rdf:type*, for example, enables the user to observe (rather than having to read and interpret) the connections and relations between different types of instances. We address the final challenge of *enabling interaction* by mimicking the browsable nature of the WWW via clickable regions, which shifts the focus to the selected resource.

We plan to carry out further usability evaluation with a larger sample and range of end users, using specific problem solving tasks to test the utility and usability of our approach. We believe that the specification of such tasks will provide the community with benchmark experiments for validating the effectiveness of methods for visualising LD. Our template-based visualisation approach utilises SPARQL queries to retrieve information from a resource's instance description and present this in a legible manner. We have detailed three static templates which demonstrate this functionality; our future work will enable the construction of templates by end users, in essence creating SPARQL queries in an implicit fashion. Providing end users with more control will also be explored, to allow for the restriction of selected property values when exploring the WoD, and filtering through only content relevant to their current activity. Finally, we plan to incorporate existing work in the field of ontology mapping, to enable templates to be loaded for instances of classes that are defined as being equivalent to others for which templates exist, e.g., loading an existing template for an instance of another class that is defined to be equivalent to `foaf:Person`.

Acknowledgements. A.-S. Dadzie is funded by the European Commission (EC) 7th Framework Programme (FP) projects *SmartProducts* and *WeKnowIt* (grants 231204, 215453). M. Rowe is funded by *WeGov* (EC FP7 248512). A.-S. Dadzie and D. Petrelli were previously funded by *X-Media* (EC FP6 026978).

References

1. Aroyo, L., et al. (eds.): ESWC 2010. LNCS, vol. 6089. Springer, Heidelberg (2010)
2. Auer, S., Doehring, R., Dietzold, S.: LESS - template-based syndication and presentation of linked data. In: Aroyo, L., et al. (eds.) [1], pp. 211–224
3. Becker, C., Bizer, C.: Exploring the geospatial semantic web with DBpedia Mobile. Journal of Web Semantics 7(4), 278–286 (2009)
4. Berners-Lee, T.: Semantic web – XML 2000 (2000), http://www.w3.org/2000/Talks/1206-xml2k-tbl
5. Berners-Lee, T., Hollenbach, J., Lu, K., Presbrey, J., Prud'ommeaux, E., Schraefel, mc.: Tabulator redux: Browsing and writing linked data. In: Linked Data on the Web Workshop at WWW '08 (2008)
6. Corlosquet, S., Delbru, R., Clark, T., Polleres, A., Decker, S.: Produce and consume linked data with Drupal! In: Bernstein, A., Karger, D.R., Heath, T., Feigenbaum, L., Maynard, D., Motta, E., Thirunarayan, K. (eds.) ISWC 2009. LNCS, vol. 5823, pp. 763–778. Springer, Heidelberg (2009)
7. Heim, P., Lohmann, S., Stegemann, T.: Interactive relationship discovery via the semantic web. In: Aroyo, L., et al. (eds.) [1], pp. 303–317
8. Herman, I., Melançon, G., Marshall, M.: Graph visualization and navigation in information visualization: A survey. IEEE Transactions on Visualization and Computer Graphics 6(1), 24–43 (2000)
9. Hirsch, C., Hosking, J., Grundy, J.: Interactive visualization tools for exploring the semantic graph of large knowledge spaces. In: Workshop on Visual Interfaces to the Social and the Semantic Web at IUI 2009 (2009)
10. Keller, T., Tergan, S.-O.: Visualizing knowledge and information: An introduction. In: Tergan, S.-O., Keller, T. (eds.) Knowledge and Information Visualization: Searching for Synergies, pp. 1–23 (2005)
11. Kerren, A., Stasko, J.T., Fekete, J.-D., North, C. (eds.): Information Visualization: Human-Centered Issues and Perspectives. Springer, Heidelberg (2008)
12. North, C.: Toward measuring visualization insight. IEEE Computer Graphics and Applications 26(3), 6–9 (2006)
13. Pietriga, E.: Semantic web data visualization with graph style sheets. In: SoftVis '06: Proc. 2006 ACM Symposium on Software Visualization, pp. 177–178 (2006)
14. Pietriga, E., Bizer, C., Karger, D.R., Lee, R.: Fresnel: A browser-independent presentation vocabulary for RDF. 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. 158–171. Springer, Heidelberg (2006)
15. Quan, D.A., Karger, R.: How to make a semantic web browser. In: WWW '04: Proc. 13th International Conference on World Wide Web, pp. 255–265 (2004)
16. Rutledge, L., van Ossenbruggen, J., Hardman, L.: Making RDF presentable: integrated global and local semantic web browsing. In: WWW '05: Proc. 14th International Conference on World Wide Web, pp. 199–206 (2005)
17. Shneiderman, B.: The eyes have it: a task by data type taxonomy for information visualizations. In: Proc. IEEE Symposium on Visual Languages, pp. 336–343 (1996)
18. Splendiani, A.: RDFScape: Semantic web meets systems biology. BMC Bioinformatics 9(suppl. 4), S6 (2008)
19. Tory, M., Moller, T.: Evaluating visualizations: do expert reviews work? IEEE Computer Graphics and Applications 25(5), 8–11 (2005)
20. Tummarello, G., Cyganiak, R., Catasta, M., Danielczyk, S., Delbru, R., Decker, S.: Sig.ma: Live views on the web of data. Journal of Web Semantics 8(4), 355–364 (2010)

Linked Data Metrics for Flexible Expert Search on the Open Web

Milan Stankovic[1,2], Jelena Jovanovic[3], and Philippe Laublet[2]

[1] hypios, 187 rue du Temple, 75003 Paris, France
[2] STIH, Université Paris-Sorbonne, 28 rue Serpente, 75006 Paris, France
[3] Université de Belgrade, Jove Ilica 154, 11000 Belgrade, Serbia
milan.stankovic@hypios.com, jeljov@fon.rs, philippe.laublet@paris-sorbonne.fr

Abstract. As more and more user traces become available as Linked Data Web, using those traces for expert finding becomes an interesting challenge, especially for the open innovation platforms. The existing expert search approaches are mostly limited to one corpus and one particular type of trace – sometimes even to a particular domain. We argue that different expert communities use different communication channels as their primary mean for communicating and disseminating knowledge, and thus different types of traces would be relevant for finding experts on different topics. We propose an approach for adapting the expert search process (choosing the right type of trace and the right expertise hypothesis) to the given topic of expertise, by relying on Linked Data metrics. In a gold standard-based experiment, we have shown that there is a significant positive correlation between the values of our metrics and the precision and recall of expert search. We also present hy.SemEx, a system that uses our Linked Data metrics to recommend the expert search approach to serve for finding experts in an open innovation scenario at hypios. The evaluation of the users' satisfaction with the system's recommendations is presented as well.

Keywords: Expert Finding, Linked Data, Linked Data Metrics, Expertise Hypothesis.

1 Introduction

Developing innovation before competitors has been the key to survival and success in many industries. Standard approaches to this include having an in-house R&D department, hiring consultants, hiring experts from expert listing websites etc. Faced with long time that those approaches take to pass from the formulation of an innovation problem to its final solution, the companies turn to open innovation [1] platforms such as hypios.com, NineSigma.com and Innocentive.com, where they post calls for solutions to their problems. Such an approach is intended to bring more diversity to the considered solutions than closed, domain-focused expert communities, and other legacy approaches could offer. Experts from various (sometimes unforeseen) domains propose solutions and the innovation seeker picks the best solutions to buy and implement.

© Springer-Verlag Berlin Heidelberg 2011

Our practice at hypios has shown that simply posting a problem online is not enough to attract diverse and innovative solutions; experts need to be invited to participate and the innovation challenge has to be introduced to them in a relevant manner. Open innovation platforms are thus faced with the challenge of identifying potential solvers – a challenge similar to but still different enough from the traditional expert search. Experts in the sense of open innovation are people who are capable of bringing a solution to a particular problem, sometimes by considering the problem from an unexpected perspective. In addition, they are not necessarily the best ranked experts for the problem topic.

Conveniently enough for the expert identification task, Web users produce more and more content, and more and more information about them is available on the Web. Those traces about users could serve as a ground for expert identification, and in fact a great number of expert search approaches has been proposed to identify and rank experts based on user traces. For instance, Buitelaar & Eigner, [2] rely on research papers, Kolari et al. [3] and Chua [4] on users' blogs, whereas Demartini [5] uses Wikipedia pages to infer and rank expertise. Content generation is not the only relevant activity for identifying experts; traces about users' bookmarking have also been used for expert finding [6], as well as question answering [7], and obtaining research grants [8].

However, the expertise search approaches found in literature often limit themselves to one or a few types of user traces and optimize their expert identification and ranking algorithms for the chosen type(s) of trace. This makes them and the systems built on top of them inflexible, in terms that they could not be easily modified or extended to accommodate a new type of user trace and/or a new kind of expertise hypothesis. In addition, their corpuses usually concern a particular domain of expertise and, accordingly, a particular expert community. As different expert communities tend to use different channels as their primary means of communication and dissemination of results, the types of representative user traces would be different for different topic of expertise. Since broad and extensible domain coverage is one of the core design principles for a Web-wide expert search system for open innovation problem, there is a strong need for versatility of such system in terms of the user traces to be used for expert identification. The state-of-the-art approaches lack this versatility as they are most often bound to a particular type of user trace due to the necessity to construct a specific data extraction approach to harvest the data from the chosen type of source.

The growing availability of user traces in structured form within the Linked Open Data (LOD) Cloud (e.g., DBLP, Semantic ChrunchBase, Twarql, RDFohloh[1] etc.) offers huge opportunities for overcoming the identified limitations of the existing expertise search approaches, as we aim to demonstrate in this paper. In particular, we propose a flexible solution for expert search in the context of open innovation problems, by leveraging the LOD Cloud and Linked Data metrics. The proposed solution helps users to select the most suitable way for identifying experts for a given topic of expertise, and subsequently applies the users' selection to the structured data about user traces on the Web (i.e., LOD) to retrieve the list of experts.

[1] Exact URIs for those data sets containing publications, company profiles, tweets, and project descriptions respectively, can be found on the official clickable LOD graph `http://richard.cyganiak.de/2007/10/lod/imagemap.html`

The rest of the paper is structured as follows: in Section 2 we formalize the notions of user traces and expertise hypotheses – the defining traits of an expert search approach. In Section 3, we introduce Linked Data metrics, and demonstrate how they can help to choose an appropriate expert search approach. Section 4 presents hy.SemEx, our proof of the concept system for expertise hypothesis recommendation and expert search on LOD. In Section 5 we present related work and conclude the paper in Section 6.

2 User Traces and Expertise Hypotheses

In this section we define the basic notions relevant to the task of finding experts by using linked data on the Web: the notion of *user trace* that represents some form of evidence that a particular user might be an expert, and the notion of *expertise hypothesis* that defines the rule by which one concludes that a particular user is an expert based on the available user traces.

2.2 The Model of User Traces on the Linked Data Web

Definition 1. *A user trace in the context of our work is an information object that can be found on the Web; it concerns a particular Web user, and is about a particular topic of expertise. The trace is a form of evidence of the expertise that the user concerned by the trace possesses in the topic that the trace is about.*

For instance, the capability of a user to write a blog post on a particular topic may be considered as an evidence of his expertise in this topic. The trace in this case is the blog post in question.

On LOD, traces are presented as <*user, trace, topic*> triples, where an information object (i.e., a trace) is related to a user (e.g., a foaf:Agent[2]) and a topic. The former relation could be the relation of creation, possession, participation (in a professional event that the trace is about), or a mention of the user in the trace, etc. The latter relation is usually formalized through the dct:subject relation pointing to a concept of general knowledge coming from a Web-based knowledge base, such as Freebase.org or DBPedia.org.

2.3 Expertise Hypothesis

Definition 2. *Expertise hypothesis is an inference mechanism, which defines how to use the information contained in a user trace to identify and/or rank experts.*

For instance, an expertise hypothesis might be: if a user has written at least 3 blog posts on topic x, he may be considered as expert on topic x. Hypotheses may be aimed at the identification of experts (like the one we just gave) or they could serve to rank experts among each other. Each legacy expert search approach relies on one or a small set of implicit expertise hypotheses. We have provided a detailed overview of expertise hypotheses from the expert search literature in [9].

[2] All the prefixes used in this paper can be dereferenced on `http://prefix.cc`

Apart from the identifying vs. ranking nature of hypotheses, they are also characterized by the type of trace they use, the level of indirection between the user and the topic of expertise, the restrictiveness, etc. In order to capture the metadata about hypotheses and allow for their meaningful exchange between expert search systems, we have created the Expertise Hypothesis Ontology (EHO)[3]. An additional important use of EHO is in describing the provenance of expert search results produced by expert finding systems. Specifically, those systems could use EHO to state which hypotheses they used to obtain the results and on which dataset. This further allows for determining whether and to what extent the results of different expert search systems could be compared and integrated. Code 1 exemplifies a hypothesis' metadata represented using EHO. A basic description includes at least a human-readable description (eho:hasTextualStatement), a rule format (eho:hasFormat) and a URL where the executable hypothesis' rule is stored (eho:hasRuleURL). In our case the rule is a SPARQL query that can retrieve experts on a given topic. The topic of choice is passed as a parameter of the query (eho:hasParameter).

```
@prefix swrc: http://swrc.ontoware.org/ontology-07#
@prefix eho: http://ontologies.hypios.com/eho#
@prefix swc: http://data.semanticweb.org/ns/swc/ontology#
@prefix dct: http://purl.org/dc/terms/

:h1 a eho:ExpertSelectionHypothesis ;
       eho:hasTextualStatement "If a user wrote a scientific publication on topic X than he
is an expert on topic X" ;
       eho:hasTraceType swrc:Publication, swrc:InProceedings, swc:Paper ;
       eho:hasRule :r1_1 .
:r1_1 a eho:ExpertiseHypothesisRule;
       eho:hasFormat :sparqlQueryFormat ;
       eho:hasRuleURL <http://hypios.com/expert-search/example-rules/rule_1_1> ;
       eho:hasParameter :p1_1.
:sparqlQueryFormat  a  eho:ExchangeFormat ;
       eho:specificationDocument <http://www.w3.org/TR/rdf-sparql-query/> .
:p1_1 a eho:RuleParameter ;
       eho:parameterType rdf:Resource ;
       eho:parameterName "expertise_topic" .
```

Code 1. An example of an expertise hypothesis metadata expressed in EHO

3 Recommendation of Expertise Hypotheses Based on Linked Data Metrics

We propose a set of Linked Data metrics aimed at guiding the expert finding process and facilitating the selection of appropriate expertise hypotheses for a given topic of expertise. The basic assumption behind our metric-based approach is that the presence and frequency of certain types of user traces as well as the presence and frequency of domain specific topics in the linked data of a particular expertise community may indicate some patterns of user behavior in that community. In other words, the constitution of linked data could reveal that certain types of traces are more significant for expertise detection related to certain topics of expertise. This assumption is based on the observed difference in the communication media that are dominant in different expertise communities. This difference becomes obvious if we look at the most recent studies of scientific communications in different fields. While biology-focused studies struggle to explain dissemination on blogs and online

[3] http://ontologies.hypios.com/eho

encyclopedia [10] and do not even try to look at Twitter, studies focused on Semantic Web research community can produce understandings of adoption of Twitter as a prominent tool in the scientific communication [11]. Being based on the above stated assumption, our Linked Data metrics enable the detection of dominant user trace types for a particular topic of expertise. Specifically, their role is to quantitatively estimate the extent of relation between a particular type of user trace and a particular topic of expertise, based on the observed constitution of the given linked dataset. Once the dominant trace type is identified for a given topic of expertise, we select expertise hypotheses that use this dominant type of trace and propose them to the user who wants to perform an expert search. The user can then select the particular expertise hypothesis of his preference, and run the expert search on a linked dataset of his choice (see Section 4).

In the following sub-sections we first motivate the introduction and use of our Linked Data metrics (Section 3.1); subsequently, we present these metrics (Section 3.2), then the experiment, which we conducted in order to evaluate the metrics (Section 3.3), and finish the section with the discussion on the scope of the metrics (Section 3.4).

3.1 Motivation

The main motivation behind our Linked Data metrics-based approach is the following:

- Take advantage of the growing number of Linked Data sources, containing diverse and constantly emerging kinds of user traces, to construct a flexible and versatile expert finding approach. Linked Data could prove especially convenient for integrating experts' contact data from different sources, as well as for taking into account the similarities between topics of expertise;
- Base the recommendation of expertise hypotheses for a given domain on the metadata (descriptions and statistics) of linked dataset(s). As opposed to the possibility of deriving overly general and potentially outdated recommendations from the analysis of the expert search literature and from qualitative studies, the metric-based approach enables recommendation that would change together with the changes of human practices and data patterns;
- In cases where the dataset for expert finding is not yet chosen, we want to provide a way to use global LOD statistics to detect global data patterns, and thus suggest the appropriate user trace types to use in the expert search approach. Based on the suggested user trace types, searching for linked open datasets that contain relevant data should be feasible.

3.2 Designing Linked Data Metrics

We have designed a number of metrics to quantify the relation between a certain user trace type and a certain topic of expertise [12]. Although a number of those metrics gave promising results in our experiments, their calculation proved to be overly time-consuming and the calculation process over multiple distributed datasets proved as almost impossible. We thus decided to take into account only the metrics that can be calculated from the dataset descriptions, without having to run additional SPARQL queries over the data. Here we mostly refer to the growing practice of providing dataset descriptions using the VoID[4] ontology and the dataset statistics using the SCOVO[5] ontology. In addition to this practice, the approaches to add data summaries to data storage systems in order to increase the performance of SPARQL queries (e.g., [13][14]) might also provide the data about the dataset composition that could be used to calculate the metrics.

3.2.1 Metrics Based on Data Quantity

The simplest imaginable metrics are based on the quantity of available data of a certain user trace type. We define Q_t to be the number of available instances of type *t*. A natural question to ask is to what this number is relative to: a dataset, or the total LOD? We call this scope of metric calculation, and it can be different in different cases of metric use. We detail the question of metric calculation scope in Section 3.4. Further on, it would be interesting to know the number of instances of a certain type satisfying some condition (e.g., having some particular concept as value of the dct:subject property). We thus define $Q_{t,\,C}$ where *C*[6] is a set of concepts (topics) that are associated with the instances to be counted. Although they may be considered as metrics themselves, Q_t and $Q_{t,C}$ serve mostly as a ground for defining more complex metrics based on topic distribution (Section 3.2.2).

3.2.2 Metrics Based on Topic Distribution

We assume that a considerable co-occurrence of particular topic with particular type of trace instances could successfully drive search for experts on the given topic. The <trace type, topic> patterns that would be found in this way could be used to privilege expertise hypotheses that rely on the particular trace type.

We define *subject homogeneity* $SH_{t,s}$ as number of user trace instances of type *t* that are associated with topic *s*, divided by the total number of user trace instances of type *t*. Subject homogeneity shows the degree of presence of the subject *s* within user traces of the type *t*. We also define *type homogeneity* $TH_{t,s}$ as number of user trace instances of type *t* that are associated with topic *s*, divided by the total number of user trace instances associated with topic *s*. This metric shows the ratio of use of particular trace type with instances relevant for a particular topic. At the same time $TH_{t,\,s}$ represents the upper bound of recall for expertise hypothesis using trace type *t* and searching for expert on topic *s*.

$$SH_{t,s} = \frac{Q_{t,s}}{Q_t} \quad TH_{t,s} = \frac{Q_{t,s}}{Q_{out:UserTrace,s}} \tag{1}$$

3.3 Measuring the Correlation Between Values of Metrics and the Performance of Expert Search

In this section we present the experiment we conducted in order to determine if there is a correlation between the values produced by our metrics and the performance of the expert search.

[4] `http://vocab.deri.ie/void/`

[5] `http://sw.joanneum.at/scovo/schema.html`

[6] When *C* is a set composed of only one topic concept *s*, then it is simply called *s*.

3.3.1 The Experimental Setting

In order to perform the experiment, we needed a dataset that would contain different types of user traces (e.g., blog and microblog posts, publications). Since we were not able to find a single LOD dataset that meets these requirements, we needed to combine data from several LOD datasets and assemble a sample dataset for testing purposes. After creating the sample dataset, we conducted a user study in order to identify in this dataset experts on a chosen set of topics. Specifically, we chose three expertise topics: dbpedia:Linked_Data, dbpedia:SPARQL, and dbpedia:Open_Data. The study ended by generating a "gold standard" for experts in the considered expertise topics. The gold standard actually consisted of a group of recognized experts for each considered topic. Having this data, we have calculated the correlation between the values produced by our Linked Data metrics (calculated based on the sample dataset constitution) and the performance of expert search that could be achieved by applying particular kinds of expertise hypotheses (on the same sample dataset).

3.3.1.1 Creation of the Sample Dataset In order to build the sample dataset we took dumps from the Semantic Web Dog Food[7] dataset, containing mostly conference and workshop publications. Additionally, we queried Sindice.com for all instances of the type sioct:BlogPost. This produced a total of 1436 instances of swrc:Publication and 837 instances of sioct:BlogPost. We used our tool for exposing semantics of tweets as linked data [15] to produce a dataset of archived tweets from the latest conferences related to the domain of Semantic Web and Web of Data. The conference archives came from the Twitter archiving service TwapperKeeper.com, and resulted in 6631 tweets in RDF. We also used the SlideShare2RDF[8] service to obtain 1657 slideshows relevant to the Web-related keywords. Such constitution of our sample dataset makes it representative for the domain of Web of Data and Semantic Web, since it contains almost all user traces (publicly available on the Web) of the mentioned trace types that were findable with a reasonable effort, on the dataset creation date which is 2010-06-15. Being domain focused, our experiment produced results that are directly applicable to the selected domains of expertise (Web of Data and Semantic Web). To make the results more broadly applicable, we intend to repeat the experiment with datasets representative for some other domains.

3.3.1.2 Data Cleaning. We faced many issues of data quality in the data retrieved from LOD, so we had to correct some data and add missing elements. In particular, the instances of sioct:BlogPost tend to be irregular. Many of them were missing the author information, so we had to construct a foaf:Agent instance based on the URL of the blog. Slideshare data was of type sioc:Item, which is quite broadly defined, so we added the bibo:Slideshow as additional type. Next problem was that only tweets produced by our system had DBpedia concepts associated with them as topics. Publications sometimes had them, but mostly had only textual topics; the same was with slideshows. None of the blog posts had them. Since our approach heavily relies on the availability of topics, we enriched the instances with topics where they were

[7] `http://data.semanticweb.org/`
[8] `http://linkeddata.few.vu.nl/slideshare/`

missing. This was done by running Zemanta[9] concept extraction service over the textual data related to instances (mostly values of dc:description, dc:title, sioc:content and swrc:abstract). The concepts that were obtained in this way were added to the instances using dct:subject property.

3.3.2 Expert Search Performance Measures

The performance of expert search is most commonly measured by precision and recall. By *precision* we understand the ratio of true positives, i.e. true experts in the total number of found expert candidates. By *recall* we understand the number of true experts found divided by the total number of true experts in a given domain. In our case determining the total number of true experts in a given domain is on the boundary of impossible, and thus we define an adapted measure of recall relative to the possibilities of a dataset (or the set of datasets) used for expert search. *Relative recall* of a particular dataset is the number of true experts found divided by the total number of true experts findable in that dataset. We consider that an expert is not findable in a dataset if the dataset's graph does not contain any path from the expert node to the node representing the topic of interest in the expert search process. Finally, since we want to favor the expert search approaches that lead to a good balance in precision and relative recall, we use the *F'*-measure that represents a harmonic mean between precision and relative recall. Our version of the F-measure gives equal importance to precision and relative recall.

$$F' = 2\frac{precision \bullet relative\_recall}{precision + relative\_recall} \tag{2}$$

3.3.3 The Gold Standard Creation

In order to evaluate the metrics in terms of their correlation with the expected performance of the expert search, we first needed to identify the true experts that may be found in our sample dataset i.e., we needed to create the gold standard. To obtain this information we have conducted a user study in which four expert users evaluated the potential experts from the sample dataset. The users were all master students at prestigious universities, who have a record of doing some important work (internship, project, bachelor thesis, etc.) on the topic of interest. The evaluation was done using an iterative process inspired by the Delphi method [16] to achieve agreement of experts about a certain opinion (most commonly a prediction). Experts give their predictions and justify them. In the second round they read the predictions and justifications of other experts, and can change their mind and thus agree on a common prediction. We used the same process to reach an agreement about the true experts on a given topic. To calculate the agreement levels between evaluators we used the kappa (*k*) agreement metric, defined in [17], to calculate the rater agreement in each phase. The *k*-statistic measures the chance corrected agreement between raters, using the confusion matrix shown in Table 1. Using these set definitions the *k*-statistic is calculated using the following formula:

$$\kappa = \frac{2(ad - bc)}{(a+c)(c+d) + (b+d)(a+b)} \tag{3}$$

[9] `http://zemanta.com`

Table 1. Rater Agreement Confusion Matrix

		Rater 1	
		Positive	Negative
Rater 2	Positive	a	b
	Negative	c	d

We have first extracted the lists of findable expert candidates for each of the three topics (dbpedia:Linked_Data, dbpedia:SPARQL, and dbpedia:Open_Data.), i.e., all expert candidates in the sample dataset for which there was any path between the expert candidate node and the topic node in the data graph. The evaluators were given these lists of candidate experts. A dereferenceable URI was provided for each expert candidate, so that the evaluators could retrieve additional information about the expert candidate and judge his expertise. Through the URI they could gain access to the expert candidate's publications, tweets, slides, blog posts, or personal homepage. Based on their judgment they assigned mark "0" or "1" to an expert candidate based on their perception of the candidate's real expertise ("1" for true experts, and "0" otherwise). The values of rater agreement statistic after the first round were very low (see the second column of Table 2). We thus conducted the second round, in which the raters could reach an agreement by correcting their results and commenting upon the other raters' marks. After the second round, the rater agreement for all three topics was above the 0.6 threshold which allowed us to take the rating from the second round as our gold standard.

Table 2. Average Rater Agreement Between Each Two Rater Pairs in the First and Second Round

	First Round	Second Round
Linked Data	0.4215	***0.6878***
SPARQL	0.4673	***0.6482***
Open Data	0.4673	***0.7228***

3.3.4 Measuring Correlation

For each type of user trace from our sample dataset, we have calculated the precision and relative recall that could be obtained if a hypothesis that uses this type of trace was applied on the sample dataset. The calculation is based on the assumption that the least restrictive hypothesis is used for a particular type (like the one accepting only one trace type to be enough for considering its author as expert).

After conducting the Pearson correlation test on the given data, we have obtained the positive values for correlation between TH and relative recall (r=0.846), between TH and F' (r=0.778), and between SH and precision (correlation coefficient r=0.619). Since all our values are above the significance threshold (r=0.576 for our sample size), we can consider the results to be statistically significant. As for the basic measures, Q_t shows no correlation with expert search performance measures, and $Q_{t,C}$ behaves like TH, just with slightly weaker correlation (r=0.777 with relative recall and r=0.761 with F'). This conclusion allows us to ground the expertise hypothesis recommendation on the values of SH and TH. In other words, based on those two metrics we would suggest the trace type that is expected to produce the best results (precision or relative recall), and the user could then choose among the hypotheses

Table 3. Values of Metrics and Expert Search performance

t	Q_t	Q_{tc}	SH	TH	precision	relative recall	F′
			Linked Data				
sioc:BlogPost	837	10	0.083	0.024	0.800	0.050	0.094
sioc:MicroblogPost	6631	96	0.014	0.236	0.469	0.296	0.363
bibo:Slideshow	1657	55	0.033	0.135	0.673	0.243	0.357
swrc:Publication	1436	86	0.060	0.211	0.721	0.408	0.521
			SPARQL				
sioc:BlogPost	837	27	0.030	0.218	0.259	0.206	0.229
sioc:MicroblogPost	6631	13	0.002	0.105	0.385	0.147	0.213
bibo:Slideshow	1657	33	0.020	0.266	0.303	0.294	0.298
swrc:Publication	1436	29	0.020	0.234	0.414	0.353	0.381
			Open Data				
sioc:BlogPost	837	5	0.006	0.013	0.600	0.025	0.048
sioc:MicroblogPost	6631	45	0.007	0.116	0.511	0.192	0.279
bibo:Slideshow	1657	80	0.048	0.207	0.300	0.200	0.240
swrc:Publication	1436	150	0.105	0.399	0.666	0.583	0.518

that rely on this trace type. Apart from expertise hypotheses suggestion, this conclusion can serve as a ground for semi-automatic hypothesis generation in which most promising user trace types would be suggested to the user who may combine them to obtain an optimal hypothesis.

3.4 The Scope of Metric Calculation

Although in our experimental setting we calculate the values of Linked Data metrics on only one dataset, it is fully imaginable to calculate them on different levels. Possible scopes of metric calculation are: (1) one dataset, (2) set of datasets and (3) the global LOD. All scopes have their own advantages, and disadvantages – many of which can be remedied.

3.4.1 Calculating the Metrics on One Dataset

Calculation of metrics for a dataset we control might be the easiest and the most accurate option, since we can normally control the generation of indexes or statistical data related to the dataset, and make sure they are accurate. A potential drawback of using this scope is that the chosen dataset might not be representative of the overall data (the portion of particular trace types in the data set might significantly differ from the global one). A possible remedy in this case is to estimate the overall structure of the LOD and then (a) make sure the dataset is representative, or (b) compose several datasets in order to reach representativeness.

3.4.2 Calculating the Metrics on a Set of Datasets

In order to calculate the metric values on the scope of several datasets, we could integrate their VoID and SCOVO descriptions to gain the knowledge about the overall composition of our chosen scope. However, at present, relying on indexes and descriptions to apply the Linked Data metrics has several potential limitations due to the data quality issues in LOD – something that has been pointed out in many sources, most notably [18] [19]. The most relevant data quality issues for our metrics are the following: 1) missing explicit type declarations of LOD instances, and 2) missing topic information of user traces contained in LOD. As shown in the creation of our sample dataset, those issues can be remedied on the dataset level, but not on the level of pre-calculated dataset descriptions that would simply reflect incomplete data. Thus relying solely on dataset descriptions to calculate the metrics on the level of the global LOD might involve some risks on accuracy, and the metric values obtained on this scope would be just approximations of real values. Dataset ranking [20] and quality filters [21] could help select the dataset descriptions to take into account and thus choose a scope that would grant higher accuracy.

3.4.3 Calculating the Metrics on the Global LOD

In cases where the dataset to work with is not chosen in advance, the global LOD might be the only choice for the scope of metrics. The same estimation strategy as for a set of datasets can be used, by relying on VOID + SCOVO descriptions. Systems[10] that automatically calculate and expose dataset descriptions could help obtain the statistics on the LOD level. The second way to perform the LOD-level estimation is to query a Semantic Web indexing service, such as Sindice.com, and obtain the number of known instances of a certain trace type, as well as traces related to a certain topic, that would allow us to calculate the metrics. This method suffers from the same dependence on the accuracy and completeness of the data in the index as reported above (Section 3.2.2).

4 hy.SemEx – The Expertise Hypothesis Recommendation and Expert Search System

Based on our findings about recommending expertise hypotheses, we have created a proof of concept system that supports a user in finding the right expertise hypothesis, and then using it to find experts. The system, currently in private alpha testing, is intended to work as a support to the Open Innovation process at hypios.com where it would allow innovation seekers to identify experts on topics of their interest and send them open innovation problems. In the following sub-sections we describe the system and present the feedback obtained from our alpha testers.

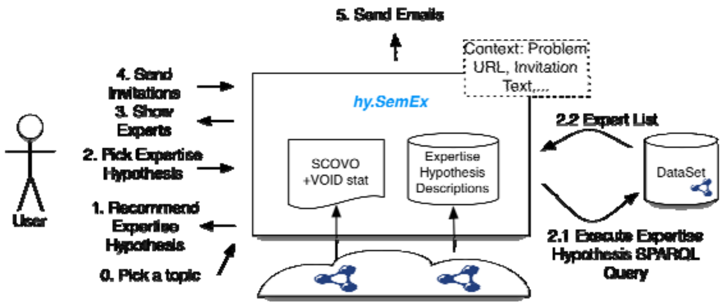

5. Send Emails
Context: Problem URL, Invitation Text,...
4. Send Invitations
3. Show Experts
2. Pick Expertise Hypothesis
1. Recommend Expertise Hypothesis
0. Pick a topic
User
hy.SemEx
SCOVO +VOID stat
Expertise Hypothesis Descriptions
2.2 Expert List
DataSet
2.1 Execute Expertise Hypothesis SPARQL Query

Fig. 1. Scenario of use of hy.SemEx

[10] `https://www.hpi.uni-potsdam.de/naumann/sites/btc2010/`

4.1 The Context and Scenario of Use

For the purpose of providing an expert search service to its clients, hypios harvests structured data containing user traces (mostly publications), and saves them in a 4store[11] triple store. In addition, a tool called SolverSurfer crawls the Web and processes free text sources containing user traces. The extracted data is then fed to the common triple store. This process makes the store resourceful for at least the domains relevant to the open innovation problem of client's interest. Hy.SemEx recommends the expertise hypotheses that client should use to find experts in this base, and allows her to run the expert search (execute the expertise hypothesis). The client can later send messages about his open innovation challenge to the identified experts. A custom designed library called SilverRabbit is in charge of sending messages over various channels of communication (tweets, e-mails, comments on slideshare presentation, etc.) depending on the known user accounts and contact data of the candidate experts. In our basic scenario of use (Figure 1), the user (innovation seeker) is offered with the list of topics, potentially relevant to her open innovation problem. The user then chooses one topic to perform the expert search and enters its DBpedia URI in the hy.SemEx system (Step 0 on Figure 1). She then expresses her preference towards a performance measure (precision or recall) and invokes the expertise hypothesis recommendation. Based on the user's choice, the appropriate metric is calculated, using the statistics about the hypios triple store as a scope of metric calculation. Using the metric values, the most appropriate user trace type is selected. The system then finds all the expertise hypotheses that use this trace type, and proposes them to the user (1) who picks one of them (2). The SPARQL query corresponding to the chosen hypothesis is then run (2.1) and the list of found experts is presented to the user (3). If the user is happy with the list, she can send it to the SilverRabbit messaging component (4), which sends the invitation (5) to the experts to come and solve the problem. The messages have previously been customized for the problem in question as a part of the open innovation campaign set-up paid by the user. Additional information about the system, such as screen-shoots, expertise hypotheses used, etc. is provided on our website[12].

4.2 Implementation Details

The system is designed for flexibility. In the core of the system is a Jena triple store that stores RDF descriptions of expertise hypotheses known to the system. New expertise hypotheses described using the EHO ontology can be easily fed to the system. The system currently works with the hypios triple store as a source for user traces and calculates the metrics based on this store as a scope. However, it is possible to reconfigure the system (in a configuration file) to calculate the metrics based on global statistics from Sindice.com. Similarly, the system can be plugged on any other data store containing traces, where it would work in a similar way it works on the hypios triple store. The endpoint information as well as dataset statistics can be fed to the system in an RDF file containing VOID and SCOVO-based descriptions and

[11] `http://4store.org/`
[12] `http://research.hypios.com/?page_id=142`

statistics[13]. In order to support statistics related to the number of traces of a certain type, as well as the number of traces with a certain topic, we have extended SCOVO with two additional properties[14]. The system is fully built in Java, using Jena to work with RDF data, and using Tapestry as a Web application framework.

4.3 Evaluation

Since the success of expert search depends on the choice and quality of dataset as much as on the choice of appropriate hypothesis, it is difficult to evaluate the quality of expertise hypothesis suggestions based on the final lists of found experts and users' satisfaction with them. For this reason we decided to directly evaluate users' satisfaction with the hypotheses suggested by the hy.SemEx system. To do that, we have conducted an evaluation study with 10 participants, and follow-up interviews with 3 of them who accepted to do it. The study participants have tested the system for 7-10 topics of their own expertise, and rated the suggested class of expertise hypothesis[15] on a scale of 1-5 (1: this class of expertise hypotheses should not be used for finding experts on this topic; 5: this is the best class for this topic). For each topic, the participants were asked to evaluate how well the proposed class of expertise hypotheses allows for finding experts on the given topic if the focus of expert search was finding: (a) the best experts, (b) as many experts as possible on the given topic. These two variations focused on: (a) precision as the desired criterion for hypothesis recommendation (where hy.SemEx used SH metric), and (b) relative recall as the recommendation criterion (corresponding to TH metric). This gave a total of 176 evaluations. The fact that the study participants chose the topics for which they have substantial knowledge, assured that they are aware of the type of communication media which is predominantly used in the community to share knowledge; this further makes their ratings reflective of the correctness of suggestions given by hy.SemEx.

The average participants' ranking for the case (a) where precision was favored was 4.234 ± 0.857 and 3.947 ± 0.751 in case (b) where recall was preferred. This result shows that suggesting expertise hypotheses to the users based on the SH and TH metrics gives reasonably good results. Specifically, the class of expertise hypothesis recommended to the participants by hy.SemEx was based on the type of user trace that was first ranked by our Linked Data metrics (SH in case (a) and TH in case (b)). We thus have shown, not only the overall correlation demonstrated in Section 3.3.4, but a considerable fitness of best-ranked type of user trace (and the corresponding class of expertise hypotheses).

After receiving suggestions, the participants could pick one of the suggested expertise hypotheses, or alternatively say they would prefer another hypotheses from a list of all hypotheses we know about[13]. The participants preferred different hypotheses for precision and different for recall. For instance, within the class of hypotheses that relied on blogs, the most preferred were the expertise hypothesis taking only authors of blog posts with more than 5 comments; however, when recall

[13] An example data set meta data file, along with a complete list of hypotheses used is provided on `http://research.hypios.com/?page_id=142`

[14] `http://ontologies.hypios.com/traceStat`

[15] We consider that the hypotheses using the same type of user trace belong the same class of hypotheses (identified by the type of user trace in question).

was demanded, they preferred the hypothesis that considered all the authors of at least two blog posts as experts. Similar patterns were observed with all classes of expertise hypotheses. In the follow-up interviews, the participants explained this by the need to use more restrictive hypotheses when precision is required. This indicates the need to include the information about restrictiveness in expertise hypothesis metadata (i.e., in the EHO ontology) and allow for automatic prioritization of hypotheses according to the user's preference towards precision or recall. Along with usability improvements, the participants also expressed a requirement for being able to change the numeric parameters of hypotheses (e.g., the number of traces taken as threshold); as well as for knowing which hypotheses were most used by others before.

5 Related Work

Although to our knowledge there are no similar approaches to use Linked Data metrics for expertise hypothesis recommendation, or for improving expert search in general, there are some similar metric-based approaches worth mentioning. The approach of Tran et al. [22] that uses data summaries to construct query routing plans for distributed query execution on LOD is conceptually very similar to ours, but applied for a different purpose. We are also not the first to use LOD as source for expert finding, but the existing approaches mostly remain mono-hypothesis i.e., limited to particular types of user traces, and even to particular topics of expertise. Saffron [23] for instance uses the Semantic Web Dog Food[16] data about publications, and augments it with research topics mined from the text, in order to allow for browsing of experts – the system does not support multiple user trace types.

6 Conclusions and Future Work

In this paper we discussed novel opportunities for expert search opened by the appearance of user traces in the form of Linked Data, most notably the opportunity of letting users choose appropriate expertise hypotheses for the relevant topic of expertise. We examined how Linked Data metrics, that reveal the constitution of a linked dataset (or set of datasets), could help to detect a good type of user trace to use for expert finding, and thus help the user prioritize those expertise hypotheses that rely on this particular type of trace. Our gold standard-based study showed that there is a significant correlation between our metrics and the expected precision and relative recall of expert search based on particular user trace types. We presented an application of this finding in a proof of concept system hy.SemEx, which uses Linked Data metrics to recommend expertise hypotheses to the user. We have evaluated the user satisfaction with the recommendations, and have found it to be very positive for the first-ranked hypothesis. In addition, the recommendation of hypothesis is not the only possible use of metrics. Approaches for helping the user construct new expertise hypothesis could also be backed up with Linked Data metrics.

In the future work we intend to repeat our experiment on a dataset that would reflect another domain and see if the same metrics still work well. Another future

[16] `http://data.semanticweb.org`

work direction is to look at other characteristics of expertise hypotheses (e.g., their restrictiveness), and not just the user trace type, and verify if there is a way to base the hypothesis recommendation on them as well. Finally, we will observe the functioning of hy.SemEx and collect the data needed to evaluate the actual expert search and not just the hypothesis recommendation part. We also intend to make the system user-friendlier, especially by making it more configurable, and by hiding the technical aspects such as concept URIs, from the user.

Acknowledgments. The work of Milan Stankovic has been partially funded by ANRT (French National Research Agency) under the grant number CIFRE N 789/2009.

References

1. Chesbrough, H.W.: Open Innovation: The New Imperative for Creating and Profiting from Technology. Harvard Business Press, Boston (2003)
2. Buitelaar, P., Eigner, T.: Topic Extraction from Scientific Literature for Competency Management. In: The 7th International Semantic Web Conference, Karlsruhe (2008)
3. Kolari, P., Finin, T., Lyons, K., Yesha, Y.: Expert Search using Internal Corporate Blogs. In: Workshop on Future Challenges in Expertise Retrieval, SIGIR 2008, pp. 2–5 (2008)
4. Chua, S.J.: Using web 2.0 to locate expertise. IBM Centre for Advanced Studies Conference (2007), Retrieved from `http://portal.acm.org/citation.cfm?id=1321250`
5. Demartini, G.: Finding experts using wikipedia. In: Proceedings of the Workshop on Finding Experts on the Web with Semantics; ISWC/ASWC 2007, Busan, South Korea (2007)
6. Noll, M.G., Yeung, C.A., Gibbins, N., Meinel, C., Shadbolt, N.: Telling Experts from Spammers: Expertise Ranking in Folksonomies. In: Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval, Boston, MA (2009)
7. Adamic, L., Zhang, J., Bakshy, E., Ackerman, M.: Knowledge sharing and yahoo answers: everyone knows something. In: Proceedings of the 17th International Conference on World Wide Web, Beijing, China, pp. 665–674. ACM, New York (2008)
8. Becerra-Fernandez, I.: Searching for experts on the web: a review of contemporary expertise locator systems. ACM Trans. on Internet Technology 6(4), 333–355 (2006)
9. Stankovic, M., Wagner, C., Jovanovic, J., Laublet, P.: Looking For Experts? What can Linked Data do for You? In: Pre-proceedings of Linked Data on the Web 2010 (LDOW) Workshop, within WWW 2010 Conference, Raleigh, NC, USA, April 26-30 (2010)
10. Comité Consultatif National d'Éthique pour les Sciences de la Vie et de la Santé (France), Communication d'informations scientifiques et médicales et société: enjeux éthiques.- Mars (2010), Retrieved from `http://www.upf.edu/pcstacademy/_docs/CCNE-Avis109.pdf`
11. Letierce, J., Passant, A., Breslin, J., Decker, S.: Understanding how Twitter is used to spread scientific messages. In: Proc. of the Web Science Conference, Raleigh, NC, USA, April 26-27 (2010)
12. Stankovic, M.: Open Innovation and Semantic Web: Problem Solver Search on Linked Data. In: Proc. of International Semantic Web Conference 2010, Shanghai, China, November 7-11 (2010)

13. Khatchadourian, S., Consens, M.: Exploring RDF Usage and Interlinking in the Linked Open Data Cloud using ExpLOD. In: Proc. Linked Data on the Web Workshop 2010 on the WWW 2010, Raleigh, NC, vol. 1430 (2010)
14. Harth, A., Hose, K., Karnstedt, M., Polleres, A., Sattler, K.: Data Summaries for On-Demand Queries over Linked Data. In: Proceedings of the 17th International Conference on World Wide Web, WWW 2010, pp. 411–420. ACM Press, Raleigh (2010)
15. Stankovic, M., Rowe, M., Laublet, P.: Mapping Tweets to Conference Talks: A Goldmine for Semantics. In: Proceeding of the Third Social Data on the Web Workshop SDoW 2010, Collocated with the International Semantic Web Conference, Shanghai, China, November 8 (2010)
16. Rowe, Wright: The Delphi technique as a forecasting tool: issues and analysis. International Journal of Forecasting 15(4) (October 1999)
17. Fleiss, J.L.: Statistical methods for rates and proportions, 2nd edn. John Wiley, NY (1981)
18. Hogan, A., Harth, A., Passant, A., Decker, S., Polleres, A.: Weaving the Pedantic Web. In: 3rd International Workshop on Linked Data on the Web (LDOW 2010) at WWW2010. CEUR Workshop Proceedings, vol. 628, CEUR-ws.org (2010)
19. Jain, P., Hitzler, P., Yeh, P.Z., Verma, K., Sheth, A.P.: Linked Data Is Merely More Data. In: Brickley, D., Chaudhri, V.K., Halpin, H., McGuinness, D. (eds.) Linked Data Meets Artificial Intelligence. Technical Report SS-10-07, pp. 82–86. AAAI Press, Menlo Park (2010)
20. Toupikov, N., Umbrich, J., Delbru, R., Hausenblas, M., Tummarello, G.: DING! Dataset Ranking using Formal Descriptions. In: WWW 2009 Workshop: Linked Data on the Web (LDOW, Madrid, Spain (2009), Retrieved from `http://sw-app.org/pub/ldow09-ding.pdf`
21. Bizer, C., Cyganiak, R.: Quality-driven information filtering using the wiqa policy framework. Journal of Web Semantics: Science, Services and Agents on the World Wide Web 7(1), 1–10 (2009)
22. Tran, T., Zhang, L., Studer, R.: Summary Models for Routing Keywords to Linked Data Sources. In: Proceedings of the 9th International Semantic Web Conferene 2010, pp. 1–16. Springer, Shanghai (2010)
23. Bordea, G.: Concept Extraction Applied to the Task of Expert Finding. In: Extended Semantic Web Conference 2010 PhD Symposium, Heraklion, Grece (2010)

Statistical Schema Induction

Johanna Völker* and Mathias Niepert

KR & KM Research Group,
University of Mannheim, Germany
{voelker,niepert}@informatik.uni-mannheim.de

Abstract. While the realization of the Semantic Web as once envisioned by Tim Berners-Lee remains in a distant future, the Web of Data has already become a reality. Billions of RDF statements on the Internet, facts about a variety of different domains, are ready to be used by semantic applications. Some of these applications, however, crucially hinge on the availability of expressive schemas suitable for logical inference that yields non-trivial conclusions. In this paper, we present a statistical approach to the induction of expressive schemas from large RDF repositories. We describe in detail the implementation of this approach and report on an evaluation that we conducted using several data sets including DBpedia.

Keywords: Linked Data, Ontologies, Association Rule Mining.

1 Introduction

The generation of ontologies from formal and semi-formal data, frequently called *Semantic Web Mining* [29], has been studied for several years within the Semantic Web community. Recently d'Amato [11] et al. suggested the term *Ontology Mining* for "all those activities that allow to discover hidden knowledge from ontological knowledge bases." This line of research is partly motivated by the crucial role ontologies play for reasoning-based applications, and by the knowledge acquisition bottleneck that is caused by the enormous efforts it takes to build highly axiomatized logical theories.

However, as argued by Auer and Lehmann [2], ontologies derived from RDF repositories can also bring major benefits for the Web of Data. Although it would be foolish to consider ontologies (or generally speaking "schemas") a panacea for all the problems currently plaguing the Web of Data, they can help to ease integration, querying and maintenance of RDF datasets. By providing conceptual descriptions of RDF graphs ontologies might facilitate, for instance, the discovery of links between disconnected data sets, or enable the detection of contradictory facts spread across the cloud of Linked Open Data. Unlike Jain et al. [18], for example, we do not believe that it will be feasible or desirable to squeeze every RDF repository under a single top-level ontology such as SUMO – and those people who are currently contributing to the success of the Linked Open Data initiative by publishing their data as RDF triples will certainly not be willing to

* Financed by a Margarete-von-Wrangell scholarship of the European Social Fund (ESF) and the Ministry of Science, Research and the Arts Baden-Württemberg.

© Springer-Verlag Berlin Heidelberg 2011

adhere to any prescribed schema. Still, if we were able to automatically generate such a schema for any given RDF repository, we would be able to provide people with formal semantics of the terminology people use for talking about their data, and possibly prepare the grounds for new types of applications.

The more RDF data becomes available the more promising seems the use of inductive, i.e. bottom-up, methods which facilitate the construction of ontologies from given facts. Inductive methods to acquiring schema-level knowledge from RDF data sources have been shown to be effective, e.g., by Lehmann et al. [17] (for an excellent overview of various types of inductive methods in the context of Linked Data see [11]). We can roughly distinguish between logical and statistical methods: While statistical methods based on conceptual clustering, for instance, tend to be more scalable and robust with respect to noisy or uncertain data, logical methods such as Inductive Logic Programming are often inferior when it comes to the generation of highly axiomatized ontologies.

In this paper, we propose an approach to generating ontologies from RDF datasets that we will refer to as *Statistical Schema Induction* (SSI). After giving a brief overview of related work (cf. Section 2), we elaborate on the theoretical foundations of our approach, including the basics of association rule mining. In Section 3, we also introduce the EL profile of OWL 2, which provides us with the logical basis for constructing ontologies that are both reasonably expressive and computationally tractable. Section 4 describes in detail the implementation of our approach, before we provide the reader with details about the experiments we conducted in order to evaluate this approach on several real-world datasets (cf. Section 5). Finally, in Section 6, we conclude with a summary and an outlook to possible future work.

2 Related Work

Our approach follows previous work in the field of ontology generation from formal and semi-formal data, e.g., in the form of RDF or OWL knowledge bases. Early approaches in this line of research rely upon systematic generalization [13] or clustering [24]. Later Grimnes et al. [14] suggested an ILP-based approach to generating descriptions of groups of people from FOAF profiles.

ILP, short for *Inductive Logic Programming*, is a type of machine learning that combines machine learning and logic programming techniques in order to derive logical theories from examples (i.e. assertions) and background knowledge. Common to all ILP-based approaches is that they adhere to the paradigm of induction – a form of inference that draws general conclusions from specific instances, assuming that the latter exemplify a general truth. ILP-based methods have successfully been applied to the problem of concept learning and ontology induction, e.g., by Cohen and Hirsh [10], but only very few implementations are commonly available one of those being the DL-Learner by Jens Lehmann [22]. In recent experiments, Hellmann et al. [17] applied the DL-Learner to several RDF knowledge bases, in order to generate definitions of classes from the YAGO ontology, for instance. Unlike our implementation, the DL-Learner uses positive and negative examples (i.e. members and non-members of these classes) randomly sampled, e.g., from DBpedia. Another particularly interesting approach has been proposed by Cimiano et al. [9], who generate intentional descriptions of the factoid answers (e.g. sets of individuals) that are returned by queries to a given knowledge base. These intentional descriptions consist of concept expressions obtained by bottom-up generalization.

Further extensional approaches to generating or refining ontologies based on given facts can be found in the area of *Formal Concept Analysis* (FCA) or Relational Exploration, respectively. OntoComP developed by Baader et al. [5] supports knowledge engineers in the acquisition of axioms expressing subsumption between conjunctions of named classes. A similar method for acquiring domain-range restrictions of object properties has been proposed later by Rudolph [27]. In both cases, hypotheses about axioms potentially missing in the ontology are generated from existing as well as from interactively acquired assertions. One of the biggest challenges for research on both FCA and ILP is uncertain and noisy input in the form of background knowledge or examples. While Auer and Lehmann [2] suggest to face this challenge by higher degrees of user interaction, we rely on statistical methods that are both robust and scalable enough to handle huge sets of Linked Data – an important prerequisite for compensating the lack of negative examples, which are not taken into account by our mining algorithms. The expressiveness of ontologies acquired by means of Statistical Schema Induction is comparable to those produced by ILP-based methods (mostly variants of $\mathcal{ALC}$) and higher than what can obtained by Relational Exploration (i.e. $\mathcal{FLE}$).

In the field of ontology learning from natural language text, we find our approach related, e.g., to methods for inducing taxonomies by means of hierarchical clustering of context vectors [8] as well as to early approaches to extracting non-taxonomic relations by *Association Rule Mining* [23]. The discovery of association rules has also been shown to facilitate the generation of ontologies from folksonomies [19] and semantic annotations in text documents [21].[1] An efficient algorithm for computing sets of association rules from RDF data was suggested by Jiang and Tan [20], while Nebot and Berlanga [25] use association rules to discover causal relations in RDF-based medical data.

Association rules have also been applied in the area of ontology matching as in the AROMA system, for example [12]. Most closely related to our approach is recent work by Parundekar et al. [26], who consider containment relationships between sets of class instantiations for producing alignments between several linked data repositories, including DBpedia. While their approach could as well be used to suggest refinements for a single ontology, they currently only acquire mappings which express subsumption or equivalence between so-called *restriction classes* roughly corresponding to $C \sqcap \exists r.D$ class expressions. In order to determine the type of correspondence between a given pair of restriction classes, Parundekar et al. rely on thresholds applied to measures of extensional overlap.

[1] Note that Association Rule Mining is similar to FCA in so far as every rule with a confidence of 1.0 directly corresponds to an implication in a formal context, and hence there has been some research on using FCA for Association Rule Mining [28].

3 Preliminaries

3.1 OWL 2 EL

The EL profile of the Web Ontology Language OWL 2 captures the expressivity of many ontologies in the life sciences and other application domains. In OWL 2 EL, which is based on the description logic $\mathcal{EL}^{++}$ [3], reasoning services such as consistency and instance checking can be performed in time that is polynomial with respect to the number of axioms. Therefore, OWL 2 EL is well-suited for applications employing ontologies that contain very large numbers of classes, properties, and axioms. Several efficient reasoning algorithms are available.

Description logics define concept descriptions inductively by a set of constructors, starting with a set $\mathsf{N_C}$ of concept (or class) names, a set $\mathsf{N_R}$ of role (or property) names, and a set $\mathsf{N_I}$ of individual names. Concept descriptions and role inclusions in $\mathcal{EL}^{++}$ are build with the constructors depicted in Figure 1. We will write a and b to denote individual names; r and s to denote role names; and C and D to denote concept descriptions. The semantics of the concept descriptions in $\mathcal{EL}^{++}$ are defined in terms of an interpretation $\mathcal{I} = (\triangle^{\mathcal{I}}, \cdot^{\mathcal{I}})$. The domain $\triangle^{\mathcal{I}}$ of this interpretation is a non-empty set of individuals and the interpretation function $\cdot^{\mathcal{I}}$ maps concepts names $A \in \mathsf{N_C}$ to a subset $A^{\mathcal{I}}$ of $\triangle^{\mathcal{I}}$, role names $r \in \mathsf{N_R}$ to a binary relation $r^{\mathcal{I}} \subseteq \triangle^{\mathcal{I}} \times \triangle^{\mathcal{I}}$, and each individual name $a \in \mathsf{N_I}$ to an individual $a^{\mathcal{I}} \in \triangle^{\mathcal{I}}$. The extension of $\cdot^{\mathcal{I}}$ to arbitrary concept descriptions is recursively defined as shown in Table 1.

Table 1. The description logic $\mathcal{EL}^{++}$ *without* nominals and concrete domains

Name	Syntax	Semantics
top	$\top$	$\triangle^{\mathcal{I}}$
bottom	$\bot$	$\emptyset$
conjunction	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
existential restriction	$\exists r.C$	$\{x \in \triangle^{\mathcal{I}} \mid \exists y \in \triangle^{\mathcal{I}} : (x,y) \in r^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$
GCI	$C \sqsubseteq D$	$C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
RI	$r_1 \circ \ldots \circ r_k \sqsubseteq r$	$r_1^{\mathcal{I}} \circ \ldots \circ r_k^{\mathcal{I}} \subseteq r^{\mathcal{I}}$

Role inclusion (RI) axioms generalize axiom types that occur often in ontology applications such as role hierarchies $r \sqsubseteq s$ and transitive roles, which can be expressed by the axiom $r \circ r \sqsubseteq r$. Also note that the bottom concept in combination with generalized concept inclusion axioms (GCIs) can be used to express disjointness of complex concept descriptions. Furthermore, it is possible to model both range and domain restrictions [4] in OWL 2 EL.

In this work, we will focus on axiom types that are captured by the EL profile of OWL 2. This way we are able to learn many of the axioms used in practical, large-scale ontologies while being able to employ efficient reasoning algorithms.

3.2 Association Rule Mining

Association rules are a very simple but useful form of implication patterns. Consider the example in Table [2]. The rows represent individuals and the columns represent the types occurring in DBpedia. A value of 1 in field (i, j) indicates that individual i is of type j. We are now interested in mining meaningful rules that provide evidence for certain axioms in the hidden schema. The framework of association rules was originally developed for large and sparse datasets such as transaction databases of international supermarket chains. A typical dataset in such a setting can have up to 10^{10} transactions (rows) and 10^6 attributes (columns). Hence, the mining algorithms developed for these applications are also applicable to the large data repositories in the open Linked Data cloud.

Let $I = \{i_1, i_2, ..., i_n\}$ be a set of n binary attributes. These attributes are usually referred to as *items*. In the open Linked Data setting items correspond to types occurring in the repository. Let $D = (t_1, t_2, ..., t_m)$ be a list of transactions (the transaction database) with each t_i a subset of I. Each transaction (that is, each row in the database) corresponds to one individual. The entry in column j has value 1 if the individual is of type i_j and 0 otherwise. The support $supp(X)$ of an itemset $X \subseteq I$ is defined as the number of transactions in the data set which contains the itemset X:

$$supp(X) = |\{t_i \in D : X \subseteq t_i\}|$$

Now, the frequent itemset mining problem is the following: Given the set I of items, a transaction database D over S, and a nonnegative threshold τ, determine the set of items whose support is at least τ. The most widely used algorithm for mining frequent itemsets is the Apriori algorithm [1].

Association rules are often used to gain a deeper understanding of the regularities and patterns of large data sets. An association rules is an implication of the form $X \Rightarrow Y$ with X and Y itemsets. While there is an exponential amount of potential association rules most state of the art algorithms take advantage of the sparsity of the transaction database and prune the search space whenever possible. In the majority of the cases, the output of frequent itemset algorithms such as Apriori is further processed to derive all association rules with a particular minimum support. The *confidence* of an association rule which is sometimes also called the *accuracy* is defined as follows:

$$conf(A \Rightarrow B) = \frac{supp(A \cup B)}{supp(A)}$$

The confidence value of an association rule can be viewed as a frequency-based maximum-likelihood estimate of the conditional probability of B occurring in the data given that A occurs in the data. The basic idea of the presented work is that association rules with a high confidence value correspond to certain OWL 2 EL axioms. For instance, a high confidence value for the association rule $A \Rightarrow B$ with A and B being RDF types would provide evidence for the validity of the subsumption axiom $A \sqsubseteq B$ because most resources that are of `rdf:type` A are also of `rdf:type` B.

Table 2. Example of a transaction database in the context of the DBpedia dataset

IRI	Comedian	Artist	Person	Airport	Building	Place	Animal
Jerry_Seinfeld	1	1	0	0	0	0	0
Black_Bird	0	0	0	0	0	0	1
Chris_Rock	1	1	1	0	0	0	0
Robin_Williams	1	0	1	0	0	0	0
JFK_Airport	0	0	0	1	1	1	0
Hancock_Tower	0	0	0	0	1	1	0
Newark_Airport	0	0	0	1	1	1	0

Example. Let us assume that the table in Figure 2 is the transaction database for a fragment of the DBpedia data set. Then, we have for instance that $supp(\{\text{Comedian, Artist}\}) = 2$, $supp(\{\text{Comedian}\}) = 3$, $supp(\{\text{Artist, Person}\}) = 1$, $supp(\{\text{Airport, Building}\} = supp(\{\text{Airport, Place}\}) = supp(\{\text{Airport, Building, Place}\}) = 2$, and $supp(\{\text{Building, Place}\}) = 3$. Furthermore, some of the association rules and their confidence values are $conf(\{\text{Comedian}\} \Rightarrow \{\text{Artist}\}) = \frac{2}{3}$ and $conf(\{\text{Airport}\} \Rightarrow \{\text{Building}\}) = \frac{2}{2} = 1.0$.

4 Statistical Schema Induction

In the following, we describe in detail our approach to inducing or enriching the schema of an RDF repository through its SPARQL endpoint. Our implementation of this approach is based on the assumption that the semantics of any RDF resource, such as a predicate for example, is revealed by patterns we can observe when considering the usage of this resource in the repository. While the general methodology of detecting such patterns by means of association rule mining can be applied to virtually any RDF repository with minor modifications, certain characteristics of a underlying RDF graph certainly facilitate the induction of a schema. We will discuss some of these characteristics in Section 5, where we detail on the experimental evaluation we conducted on different datasets.

The overall process of SSI can be summarized as follows (cf. Figure 1):

1. First, we acquire the *terminology*, i.e. the non-logical vocabulary of the OWL ontology to be constructed, by posing SPARQL queries to the repository's endpoint (cf. Section 4.1). The result of this step is a set of relational database tables containing the URIs of all those RDF resources which we assume to correspond to classes and properties.[2] Note that we also assign unique identifiers to certain combinations of resources (e.g. any pair of two predicates r_1 and r_2) as we would like use those for building complex class or property expressions (e.g. $r_1 \circ r_2$).

[2] In order to identify potential classes and properties in an RDF graph, we rely on heuristics similar to those suggested by Bechhofer and Volz [6] and taken up later by Hellmann et al. [17].

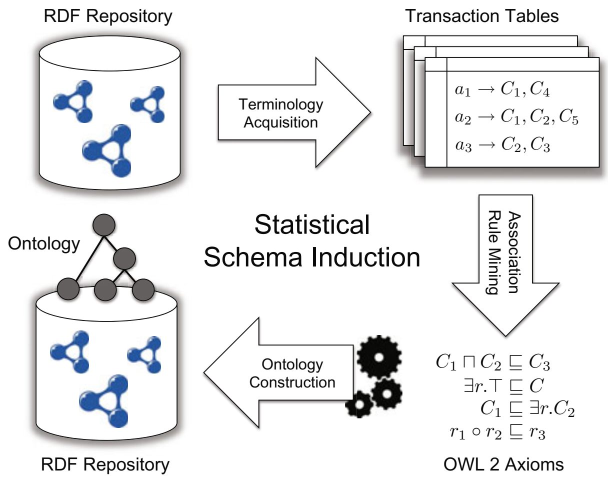

RDF Repository
Transaction Tables
Terminology Acquisition
$a_1 \rightarrow C_1, C_4$
$a_2 \rightarrow C_1, C_2, C_5$
$a_3 \rightarrow C_2, C_3$
Ontology
Statistical Schema Induction
Association Rule Mining
Ontology Construction
$C_1 \sqcap C_2 \sqsubseteq C_3$
$\exists r.\top \sqsubseteq C$
$C_1 \sqsubseteq \exists r.C_2$
$r_1 \circ r_2 \sqsubseteq r_3$
RDF Repository
OWL 2 Axioms

Fig. 1. Worfklow of the Statistical Schema Induction framework

2. Second, we construct the *transaction tables*, which we need in order to mine the dataset for the various kinds of OWL axioms (cf. Section 4.2). Each transaction, i.e. row in a transaction table, corresponds to a resource or a pair of resources, respectively. While for every single resource the items in a transaction are named or complex "classes" according to our terminology, a pair of resources always maps to the set of predicates or predicate chains (i.e. "property expressions") they are linked with. We then mine the transaction tables for *association rules*.
3. Finally, every association rule gets translated into an OWL 2 EL axiom. The support and the confidence values of the rules are taken into account when the ontology is constructed in a fully automatic manner (cf. Section 4.3).

4.1 Terminology Acquisition

Named classes. Initially, we gather information about those resources which are likely to represent *classes* $C \in \mathsf{N}_\mathsf{C}$ in the ontology that we would like to generate. We do so by means of a SPARQL query motivated by a simple heuristic: every object of an `rdf:type` statement is a class, whereas the subjects of all such statements provide us with the instances of these classes.[3] Since the syntax and semantics of RDF do not constrain the use of `rdf:type` statements nor otherwise enforce a division into assertional and terminological level, we cannot expect this heuristic to work for every RDF graph. However, as we will see in Section 5, it yields good results for several of the most well-known datasets. Every resource supposed to be a class or an individual gets assigned a unique numerical identifier and is stored in a relational database.

[3] We only consider explicit `rdf:type` statements, hence ignore those which are entailed e.g. by `owl:sameAs` statements, which often have an unclear semantics [16].

While the names of the "individuals" (i.e. those resources explicitly stated to have an `rdf:type`) are not relevant for our approach as the individuals anyway will not become part of the schema, we have to be able to uniquely identify them in order to construct the transactions table (see further below). In cases where the overall number of resources considered individuals is too high for storing them locally, one should consider the use of sampling heuristics (as suggested, e.g., by d'Amato et al. [11]) or of a Map-Reduce infrastructure on top of a distributed file system. The latter would actually work very well with our approach as the computation of the transactions tables from the data gathered at this stage can be conducted in a completely parallel way.

Object properties. In a similar way, we collect the names of all those RDF resources which we assume to represent *object properties* $r \in \mathsf{N_R}$ in the ontology: Every predicate of an RDF triple which belongs to the DBpedia namespace and whose object is linked to another resource by means of an `rdf:type` statement is considered an object property. Again, we store both, all of these predicates and the unique pairs of resources linked by any of these predicates, in our database.

Class expressions. Now that we have acquired the basic terminology, i.e. the names of all resources to denote named classes or properties, we can turn to complex class and property expressions. Since we want to be able to also mine the dataset for domain and range restrictions such as $\exists r.\top \sqsubseteq C$, for example, we have to assign unique identifiers to the following types of class expressions: $\exists r.C$, $\exists r.\top$ and $\exists r^{-1}.\top$ for each $r \in \mathsf{N_R}$ and $C \in \mathsf{N_C}$.[4] Note that there can be resources which are not explicitly stated to be of some type (cf. named classes) while still fitting some of these class expressions (e.g. because the respective resource is linked to another one by virtue of a property r). For this reason, we also need to extend our initially computed set of identifiers for potential individuals.

Property chains. Finally, as motivated in Section 3.1, we would like to acquire *transitivity* axioms for all the predicates (i.e. potential object properties) in the dataset. Transitivity can be expressed by a particular type of property chain inclusion axiom, namely $r \circ r \sqsubseteq r$ for $r \in \mathsf{N_R}$. Therefore, we again create a database table for mapping each property chain expression to a unique identifier and, similarly as for the class expressions, assign a new identifier to a pair of resources whenever this pair is not linked directly by any object property but only by a property chain of the aforementioned shape.

4.2 Association Rule Mining

Before we can start mining for association rules as described in Section 4.3, we have to create *transaction tables* for the various types of axioms that we would like to become part of the ontology. Figure 3 gives an overview of the types of axioms covered by our current implementation and the corresponding transaction tables. For example, axioms of the type $C \sqsubseteq D$ (that is, those expressing subsumption between atomic classes) can be mined from a transaction table whose

[4] Range restrictions can be expressed in OWL 2 EL, even though inverse properties are not included in the profile [4].

Table 3. Each row in a transaction table either corresponds to a resource a or pair of resources (a, b) for $a, b \in \mathsf{N_I}$, which are assumed to represent an individual or a pair of individuals, respectively. We output the identifier of C iff a is linked to a resource named C by means of an `rdf:type` statement, and the identifier of r iff a and b are connected by the predicate r. Likewise, if a row in the transaction table contains the identifier of a class expression $\exists r.C$, for example, this means that the corresponding resource a is linked to another resource of `rdf:type` C by virtue of r.

Axiom Type	Transaction Table	Association Rule
$C \sqsubseteq D$	$a \rightarrow C_1, ..., C_n$ for $a \in \mathsf{N_I}$	$\{C_i\} \Rightarrow \{C_j\}$
$C \sqcap D \sqsubseteq E$	$a \rightarrow C_1, ..., C_n$ for $a \in \mathsf{N_I}$	$\{C_i, C_j\} \Rightarrow \{C_k\}$
$D \sqsubseteq \exists r.C$	$a \rightarrow C_1, ..., C_l, \exists r_1.C_{11}, ..., \exists r_m.C_{mn}$ for $a \in \mathsf{N_I}$	$\{C_k\} \Rightarrow \{\exists r_j.C_{jk}\}$
$\exists r.C \sqsubseteq D$	$a \rightarrow C_1, ..., C_l, \exists r_1.C_{11}, ..., \exists r_m.C_{mn}$ for $a \in \mathsf{N_I}$	$\{\exists r_j.C_{jk}\} \Rightarrow \{C_i\}$
$\exists r.\top \sqsubseteq C$	$a \rightarrow C_1, ..., C_l, \exists r_1.\top, ..., \exists r_m.\top$ for $a \in \mathsf{N_I}$	$\{\exists r_j.\top\} \Rightarrow \{C_i\}$
$\exists r^{-1}.\top \sqsubseteq C$	$a \rightarrow C_1, ..., C_l, \exists r_1^{-1}.\top, ..., \exists r_m^{-1}.\top$ for $a \in \mathsf{N_I}$	$\{\exists r_j^{-1}.\top\} \Rightarrow \{C_i\}$
$r \sqsubseteq s$	$(a, b) \rightarrow r_1, ..., r_n$ for $(a, b) \in \mathsf{N_I} \times \mathsf{N_I}$	$\{r_i\} \Rightarrow \{r_j\}$
$r \circ r \sqsubseteq r$	$(a, b) \rightarrow r_1, ..., r_n, r_1 \circ r_1, ...r_n \circ r_n$ for $(a, b) \in \mathsf{N_I} \times \mathsf{N_I}$	$\{r_i \circ r_i\} \Rightarrow \{r_i\}$

rows correspond to those individuals having at least one `rdf:type` $C \in \mathsf{N_C}$. Each row in this table contains the identifiers of those classes $C_1, ..., C_n$ the respective individual belongs to, and can be determined by a simple SPARQL query (`SELECT distinct ?c WHERE <`a`> a ?c`) followed by a lookup in the previously built terminology tables (cf. Section 4.1). Note that axioms that involve the same types of class or property expressions can be mined from the same transaction tables. For instance, both the $D \sqsubseteq \exists r.C$ and the $\exists r.C \sqsubseteq D$ axioms are mined from the same transaction database. Thus, we only need 6 transaction databases in order to mine the dataset for the axiom types listed in Table 3.

4.3 Ontology Construction

As indicated by Figure 3, the association rules mined from the various transaction tables can be translated into OWL 2 EL axioms in a relatively straightforward way. The confidence and support value for each of the association rules provides us with a measure of certainty, which we take into account when constructing the ontology.[5]

Following [15], we pursue a simple greedy strategy for adding the acquired axioms to an initially empty or to an existing OWL ontology that we would like to refine: First, we sort all of the generated axioms in descending order based on their certainty values. Then we add them to the ontology one by one, checking the coherence of the ontology after the addition of each axiom. In case any of the classes in the ontology becomes incoherent, e.g. because an axiom states disjointness

[5] By setting a confidence threshold of 1.0 we obtain an ontology that perfectly fits the data, i.e. which does not contain any inconsistencies if merged with the factoid knowledge in the RDF repository.

between two classes which subsume each other according to a previously added axiom, we retract the most recent axiom and continue with the next one.

5 Evaluation

In the following, we report on several experiments we conducted in order to evaluate Statistical Schema Induction with real-world data. The results of these experiments are available online and can be downloaded from a dedicated web page.[6] By the first experiment based on the DBpedia dataset (cf. Section 5.1) we were aiming to gain some insights regarding the quality of the generated ontologies as compared to existing, manually constructed schemas. The comparatively large size of the DBpedia dataset makes it a good benchmark for the scalability of our implementation. In a second experiment (see Section 5.2), we assessed the feasibility of our approach on a set of smaller RDF datasets, all of them taken from `data.gov.uk`.

Both of these experiments were carried out on a an AMD 64bit DualCore computer with 2,792 MHz and 8 GB RAM. The Java-based implementation of our approach makes use of various publicly available libraries for database access (MySQL 5.0.51), ontology management (Pellet 2.2.1 and OWL API 3.0.0) and Linked Data querying (Jena 2.6.3). In addition, we applied the Apriori implementation by Borgelt and Kruse [7] to mine the association rules.

```
1110   1325   6293   0      1      144
4065   4665   4695   1146   6330   6973   64     185
1146   6330   6973   64     68     141
3235   6668   6769   3242   5049   6673   3907   2    66
1110   1325   6293   0      73     144
```

Fig. 2. Textual serialization of a transaction table

The input to this implementation were textual serializations of the transaction tables (cf. Table 3). Figure 2 shows an excerpt from an input file that enables us to acquire axioms of the form $C \sqsubseteq \exists r.D$ and $\exists r.C \sqsubseteq D$ for the DBpedia dataset (cf. Section 5.1). The items in each transaction like `144` (*Place*) or `3907` ($\exists language.Language$), for example, are the identifiers of those named or complex classes the respective individual belongs to. Note that we only need to compute transactions for those individuals which are known to be members of at least one named or complex class. For the DBpedia dataset, the number of rows in the various transaction tables varied between 5,217,133 (transitivity axioms) and 1,477,796 (domain and range restrictions).

From the computed association rules, we generated the types of axioms listed by Table 3. Moreover, we generated disjointness axioms ($C \sqcap D \sqsubseteq \bot$) between classes with more than 100 instances that do not have any individuals in common. The confidence values for these axioms were generated by normalizing the product of the number of instances for each pair of classes.

[6] `http://code.google.com/p/gold-miner/`

τ	# axioms	recall	precision	F_1 score
1.0	365	0.997	0.992	0.995
0.9	373	0.997	0.971	0.983
0.8	381	0.997	0.950	0.973

(a) Without support threshold

τ	# axioms	recall	precision	F_1 score
1.0	339	0.926	0.991	0.957
0.9	347	0.926	0.968	0.947
0.8	354	0.926	0.949	0.937

(b) With support threshold of 10

Fig. 3. Recall, precision and F_1 values for **subsumption axioms** between atomic classes for varying thresholds on the confidence values

τ	# axioms	recall	precision	F_1 score
1.0	950	0.900	0.808	0.852
0.9	1143	0.946	0.655	0.774
0.8	1181	0.946	0.683	0.793

(a) Without support threshold

τ	# axioms	recall	precision	F_1 score
1.0	821	0.790	0.821	0.805
0.9	1036	0.835	0.576	0.682
0.8	1092	0.838	0.558	0.670

(b) With support threshold of 10

Fig. 4. Recall, precision and F_1 values for **domain restriction axioms** for varying thresholds on the confidence values

5.1 DBpedia

We evaluated the generated ontology by comparing it to the DBpedia ontology[7] (version 3.5.1), which we considered the most natural gold standard. The DBpedia ontology was created by a manual mapping of 1,055 Wikipedia infobox templates to 259 named classes. Besides these classes, the ontology comprises 602 object properties, 674 datatype properties, 257 explicit subsumption axioms as well as 459 domain and 482 range restrictions. 1,477,796 of the roughly 3.4 million "things" (i.e. RDF resources representing Wikipedia articles) are explicitly classified with regard to the DBpedia ontology.

However, as the expressivity of the DBpedia ontology is relatively low (it equals the complexity of the $\mathcal{ALF}$(D) description logic), we only considered those types of axioms that are common to both ontologies when comparing the two schemas: subsumption between named classes and property restrictions. Tables 3, 4, and 5 list the results of the schema induction process for various thresholds on the confidence values. The time needed to compute the association rules was less than 5 seconds for the largest transaction table, which confirms the scalability of the Apriori algorithm to the large Linked Data repositories. The recall and precision scores are computed *relative* to the DBpedia ontology. Hence, not all false positives and false negatives with respect to the DBpedia ontology would necessarily be incorrect in terms of a more complete gold standard. Note that for the comparison of the two ontologies we did not only consider the explicit axioms, but all the *inferable* class subsumption and property restriction axioms.

[7] `http://wiki.dbpedia.org/Ontology`

τ	# axioms	recall	precision	F_1 score
1.0	401	0.392	0.666	0.494
0.9	740	0.712	0.655	0.682
0.8	868	0.790	0.620	0.695

(a) Without support threshold

τ	# axioms	recall	precision	F_1 score
1.0	206	0.258	0.854	0.396
0.9	740	0.580	0.512	0.544
0.8	840	0.658	0.604	0.630

(b) With support threshold of 10

Fig. 5. Recall, precision and F_1 values for **range restriction axioms** for varying thresholds on the confidence values

5.2 Other Datasets

In order to test the applicability of Statistical Schema Induction to RDF datasets other than DBpedia (in particular to datasets that do not come with any existing schema yet), we performed a second evaluation experiment based on the RDF repository of `data.gov.uk`. In this experiment, we focused on the taxonomy of named classes and merely generated axioms of the form $C \sqsubseteq D$. Due to time constraints and the lack of a proper gold standard for this dataset we only report our most important findings, and refer the reader to the aforementioned website that we setup for our experiments.

Without any changes to our implementation, we were able to compute appropriate transaction tables for five subsets of `data.gov.uk` each of these subsets corresponding to a public sector:[8] *reference*, *eduction*, *ordnance*, *transport* and *finance*. For three of them we obtained a proper class hierarchy,[9] while closer inspection of the other two datasets (*ordnance* and *finance*) revealed that none of the resources in the dataset was stated to be of more than one `rdf:type`.

One might argue that the redundancy caused by multiple type statements cannot be assumed to be common in real-world datasets. However, note that the kind of axioms involving complex class expressions or any of the property subsumption axioms could still be acquired in this case. Moreover, whenever we do not find multiple `rdf:type` statements for all of the RDF resources (e.g. in the case of the *ordnance* dataset of `data.gov.uk` only very few resources have more than one type), we could pursue the following bootstrapping-like strategy: First, we mine the dataset for domain-range restrictions of the predicates that we assume to represent object properties. For example, we might find the domain of a certain predicate r to be of type C, that is $\exists r.\top \sqsubseteq C$. Then, we use these restrictions in order to "classify" all of the resources in the RDF graph. In particular, adhering to the OWL semantics of domain-range restrictions, we can infer that each resource that has the property r must be of type C and likewise for the range. Those additional type statements could finally help to induce the hierarchy of named classes.

[8] As the *legislation* part of `data.gov.uk` uses only a very limited set of identifiers for the objects of `rdf:type` statements, we were unable to acquire a sufficiently rich terminology for this dataset.

[9] For example, the axioms *DeputyDirector* $\sqsubseteq$ *CivilServicePost* and *MinisterialDepartment* $\sqsubseteq$ *Department* were mined from the *reference* part of the `data.gov.uk` dataset.

Without applying this strategy, we obtained 64 classes and 47 axioms for *education*, 62 classes and 137 axioms for *transport*, 20 classes and 17 axioms for *reference*, 29 classes and 3 axioms for *ordnance*, as well as 41 classes and 0 axioms for *finance*, where "axioms" refers to explicit subsumption axioms ($C \sqsubseteq D$).

6 Conclusion and Future Work

In this paper, we introduced statistical schema induction as a means to generating ontologies from RDF data. Our approach based on association rule mining has been tested on several real-world datasets. While the first results are actually very promising, we are well aware of the fact that our implementation could be improved in various respects.

As future work we envision, for example, an adaptation of our approach to more expressive description logics. In particular, we will extend our implementation to also capture property disjointness ($r \sqsubseteq \neg s$), inverse properties ($r \equiv s^{-1}$) and cardinality restrictions (e.g. $C \sqsubseteq\ \leq 1 r.\top$). Furthermore, we would like to facilitate a more efficient construction of the transaction tables by appropriate sampling strategies or a Map-Reduce framework for distributed computation. Another very promising way to increase the scalability of our approach could be the use of incremental methods for adapting a generated ontology to subsequent changes in the underlying dataset. As long as we can assume these changes to be strictly monotonic, the necessary adaptations to the transaction tables will be linear in time, and efficient algorithms for mining association rules could suggest appropriate ontology refinements within a few seconds at most. It is thus tempting to imagine, for example, an on-the-fly refinement of the DBpedia ontology that keeps it synchronized with the DBpedia live dataset. Finally, we are confident that these optimizations as well as existing instance mapping techniques will facilitate the application of Statistical Schema Induction across even larger and more heterogenous fragments of the Linked Data cloud.

Acknowledgements. We especially thank Jens Lehmann and the AKSW research group for their invaluable advice and technical support.

References

1. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules in large databases. In: Proceedings of 20th International Conference on Very Large Data Bases, pp. 487–499. Morgan Kaufmann, San Francisco (1994)
2. Auer, S., Lehmann, J.: Creating knowledge out of interlinked data. Semantic Web 1(1-2), 97–104 (2010)
3. Baader, F., Brandt, S., Lutz, C.: Pushing the $\mathcal{EL}$ envelope. In: Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence IJCAI, Edinburgh, UK. Morgan-Kaufmann Publishers, San Francisco (2005)
4. Baader, F., Brandt, S., Lutz, C.: Pushing the EL envelope further. In: Clark, K., Patel-Schneider, P.F. (eds.) Proceedings of the OWLED 2008 DC Workshop on OWL: Experiences and Directions (2008)

5. Baader, F., Ganter, B., Sertkaya, B., Sattler, U.: Completing description logic knowledge bases using formal concept analysis. In: Veloso, M.M. (ed.) Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI), pp. 230–235. AAAI Press, Menlo Park (2007)
6. Bechhofer, S., Volz, R.: Patching syntax in OWL ontologies. In: McIlraith, S.A., Plexousakis, D., van Harmelen, F. (eds.) ISWC 2004. LNCS, vol. 3298, pp. 668–682. Springer, Heidelberg (2004)
7. Borgelt, C., Kruse, R.: Induction of association rules: Apriori implementation. In: 15th Conference on Computational Statistics, pp. 395–400 (2002)
8. Cimiano, P., Hotho, A., Staab, S.: Comparing conceptual, divise and agglomerative clustering for learning taxonomies from text. In: Proceedings of the 16th European Conference on Artificial Intelligence (ECAI), pp. 435–439. IOS Press, Amsterdam (2004)
9. Cimiano, P., Rudolph, S., Hartfiel, H.: Computing intensional answers to questions – an inductive logic programming approach. Data & Knowledge Engineering 69(3), 261–278 (2010)
10. Cohen, W.W., Hirsh, H.: Learning the classic description logic: Theoretical and experimental results. In: Proceedings of the 4th International Conference on Principles of Knowledge Representation and Reasoning (KR), pp. 121–133. Morgan Kaufmann, San Francisco (1994)
11. d'Amato, C., Fanizzi, N., Esposito, F.: Inductive learning for the semantic web: What does it buy? Semantic Web 1(1-2), 53–59 (2010)
12. David, J., Guillet, F., Briand, H.: Association rule ontology matching approach. International Journal on Semantic Web and Information Systems 3(2), 27–49 (2007)
13. Delteil, A., Faron-Zucker, C., Dieng, R.: Learning ontologies from rdf annotations. In: Maedche, A., Staab, S., Nedellec, C., Hovy, E.H. (eds.) Proceedings of the 2nd Workshop on Ontology Learning (OL) at the 17th International Conference on Artificial Intelligence (IJCAI). CEUR Workshop Proceedings, vol. 38, CEUR-WS.org (2001)
14. Grimnes, G.A., Edwards, P., Preece, A.D.: Learning meta-descriptions of the FOAF network. In: McIlraith, S.A., Plexousakis, D., van Harmelen, F. (eds.) ISWC 2004. LNCS, vol. 3298, pp. 152–165. Springer, Heidelberg (2004)
15. Haase, P., Völker, J.: Ontology learning and reasoning — dealing with uncertainty and inconsistency (ISWC International Workshop, URSW 2005-2007, Revised Selected and Invited Papers). In: da Costa, P.C.G., d'Amato, C., Fanizzi, N., Laskey, K.B., Laskey, K.J., Lukasiewicz, T., Nickles, M., Pool, M. (eds.) URSW 2005 - 2007. LNCS (LNAI), vol. 5327, pp. 366–384. Springer, Heidelberg (2008)
16. Halpin, H., Hayes, P.J., McCusker, J.P., McGuinness, D.L., Thompson, H.S.: When owl:sameAs isn't the same: An analysis of identity in 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. 305–320. Springer, Heidelberg (2010)
17. Hellmann, S., Lehmann, J., Auer, S.: Learning of OWL class descriptions on very large knowledge bases. International Journal on Semantic Web and Information Systems 5(2), 25–48 (2009)
18. Jain, P., Hitzler, P., Yeh, P.Z., Sheth, A.P.: Linked data is merely more data. In: Proceedings of the AAAI Spring Symposium, Linked AI: Linked Data Meets Artificial Intelligence (2010)
19. Jäschke, R., Hotho, A., Schmitz, C., Ganter, B., Stumme, G.: Discovering shared conceptualizations in folksonomies. Journal of Web Semantics 6(1), 38–53 (2008)

20. Jiang, T., Tan, A.-H.: Mining RDF metadata for generalized association rules. In: Bressan, S., Küng, J., Wagner, R. (eds.) DEXA 2006. LNCS, vol. 4080, pp. 223–233. Springer, Heidelberg (2006)
21. Kuittinen, H., Tuominen, J., Hyvönen, E.: Extending an ontology by analyzing annotation co-occurrences in a semantic cultural heritage portal. In: Proceedings of the Workshop on Collective Intelligence (ASWC-CI) at the 3rd Asian Semantic Web Conference, ASWC (2008)
22. Lehmann, J.: DL-Learner: learning concepts in description logics. Journal of Machine Learning Research (JMLR) 10, 2639–2642 (2009)
23. Mädche, A., Staab, S.: Discovering conceptual relations from text. In: Horn, W. (ed.) Proceedings of the 14th European Conference on Artificial Intelligence (ECAI), pp. 321–325. IOS Press, Amsterdam (2000)
24. Maedche, A., Zacharias, V.: Clustering ontology-based metadata in the semantic web. In: Elomaa, T., Mannila, H., Toivonen, H. (eds.) PKDD 2002. LNCS (LNAI), vol. 2431, pp. 348–360. Springer, Heidelberg (2002)
25. Nebot, V., Berlanga, R.: Mining association rules from semantic web data. In: García-Pedrajas, N., Herrera, F., Fyfe, C., Benítez, J.M., Ali, M. (eds.) IEA/AIE 2010. LNCS, vol. 6097, pp. 504–513. Springer, Heidelberg (2010)
26. 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)
27. Rudolph, S.: Acquiring generalized domain-range restrictions. In: Medina, R., Obiedkov, S. (eds.) ICFCA 2008. LNCS (LNAI), vol. 4933, pp. 32–45. Springer, Heidelberg (2008)
28. Stumme, G.: Efficient data mining based on formal concept analysis. In: Hameurlain, A., Cicchetti, R., Traunmüller, R. (eds.) DEXA 2002. LNCS, vol. 2453, pp. 534–546. Springer, Heidelberg (2002)
29. Stumme, G., Hotho, A., Berendt, B.: Semantic web mining: State of the art and future directions. Journal of Web Semantics 4(2), 124–143 (2006)

SIHJoin: Querying Remote and Local Linked Data

Günter Ladwig and Thanh Tran

Institute AIFB, Karlsruhe Institute of Technology, Germany
{guenter.ladwig,ducthanh.tran}@kit.edu

Abstract. The amount of Linked Data is increasing steadily. Optimized top-down Linked Data query processing based on complete knowledge about all sources, bottom-up processing based on run-time discovery of sources as well as a mixed strategy that combines them have been proposed. A particular problem with Linked Data processing is that the heterogeneity of the sources and access options lead to varying input latency, rendering the application of blocking join operators infeasible. Previous work partially address this by proposing a non-blocking iterator-based operator and another one based on symmetric-hash join. Here, we propose *detailed cost models* for these two operators to systematically compare them, and to allow for query optimization. Further, we propose a novel operator called the *Symmetric Index Hash Join* to address one open problem of Linked Data query processing: to query not only remote, but also local Linked Data. We perform experiments on real-world datasets to compare our approach against the iterator-based baseline, and create a synthetic dataset to more systematically analyze the impacts of the individual components captured by the proposed cost models.

1 Introduction

The amount of Linked Data on the Web is large and ever increasing. This development is exciting, paving new ways for next generation applications on the Web. We contribute to this development by investigating the problem of how to process queries against Linked Data.

Linked Data query processing can be seen as a special case of federated query processing, i.e., to process queries against data that reside in different data sources. However, the highly distributed structure and evolving nature of Linked Data presents unique challenges. In particular, as discussed in [6], the number of Linked Data sources is large (*volume*); sources evolve quickly (*dynamic*); sources vary in size, there is no standard for source descriptions, and access options vary (*heterogeneity*). As source descriptions, Harth et al. [2] proposed a probabilistic data structure to capture and store locally rich statistics about remote sources, used to determine relevant sources and to optimize query processing. Hartig et al. [3] proposed a method for dealing with the dynamic aspect of Linked Data query processing. As opposed to [2], the strategy employed here does not rely on complete knowledge about Linked Data available as source descriptions but is based on run-time source discovery via URI lookups. In previous work [6], we

© Springer-Verlag Berlin Heidelberg 2011

proposed a mixed strategy that is able to leverage locally stored source descriptions if they exist, and to discover other sources at run-time.

Partially, our previous work also elaborates on the aspect of *join implementation*. In this setting, standard blocking iterator-based join is not optimal due to possibly very high network latency (as a result of Linked Data heterogeneity). Instead of blocking, Hartig et al. [3] proposed a non-blocking iterator-based join operator (NBIJ) that employs a busy-waiting strategy. Basically, it is a workaround that temporarily rejects inputs when the other inputs needed for the join are not available. In previous work [6], we discussed a conceptually cleaner strategy called push- and stream-based query processing based on the symmetric hash join (SHJ) [15], where source data is treated as finite streams that can arrive at any time in any order. In theory, this strategy is better suited to deal with network latency as it is driven by incoming data (i.e., push- instead of pull-based) and thus, does not require temporary input rejection. However, a systematic comparison of these two strategies at the conceptual level as well as experimental results that can validate possible differences are missing.

Another problem is that so far, joins are performed on remote data, but in practice, Linked Data (can be imported and) might be available locally, giving rise to non-blocking operators capable of *processing both remote and local data.*

Contributions. In this work we focus on join operators:

– We propose a new join operator called Symmetric Index Hash Join (SIHJ) that is non-blocking, pushed-based, stream-based, and in particular, is able to process both remote and local linked data.
– We propose a cost model that can be used to analyze this operator given only remote data, only local data, or a combination of them. Further, we provide a cost model for the proposed NBIJ [3]. These two cost models can be used for query optimization, and allow us to compare the mechanisms underlying these operators in a systematic fashion.
– In an experimental comparison, we evaluate these two approaches on real-world datasets and a synthetic dataset to more systematically analyze the impacts of the individual components captured by the proposed cost models.

Outline. In Section 2 we introduce Linked Data query processing and motivate our approach. Section 3 presents and analyzes the symmetric index hash join operator. We compare the SIHJ operator to the previously proposed NIHJ in Section 4. Finally, we present related work in Section 5, before discussing the evaluation results in Section 6 and the conclusions in Section 7.

2 Preliminaries

Linked Data Query Processing. As usual [3,6], we conceive Linked Data sources as interlinked sets of RDF triples [5]:

Definition 1. *A* source *d is a set of RDF triples $\langle s^d, p^d, o^d \rangle \in T^d$ where s^d is called the subject, p^d the predicate and o^d the object. There is a function ID* *which associates a source d with a unique URI. There is a* link *between two sources d_i and d_j if the URI of d_i appears as the subject or object in at least one triple of d_j, i.e., $\exists t \in T^{d_j} : s^d(t) = ID(d_i) \vee o^d(t) = ID(d_i)$; or vice versa, i.e., $\exists t \in T^{d_i} : s^d(t) = ID(d_j) \vee o^d(t) = ID(d_j)$ (then d_i and d_j are said to be* interlinked*). The union set of interlinked sources $d_i \in D$ constitutes the* Linked Data $T^D = \{t|t \in T^{d_i}, d_i \in D\}$.

The standard language for querying RDF data is SPARQL [10]. An important part of SPARQL queries are basic graph patterns (BGP). Work on Linked Data query processing so far focused on the task of answering BGP queries. In this work we focus on BGP queries that form a connected graph, i.e., can be answered without cross products:

Definition 2. *A* connected basic graph pattern *q is a set of* triple patterns *$\langle s^q, p^q, o^q \rangle \in T^q$ where every s^q, p^q and o^q is either a* variable *or a* constant*. There are some variables appearing in several patterns $t^q \in T^q$ such that together, the set of patterns T^q forms a connected graph.*

Since Linked Data triples in T^D also form a graph, processing queries in this context amounts to the task of graph pattern matching. In particular, an *answer* (also called a *solution mapping* or *query binding*) to a BGP query is given by μ which maps the variables in query graph pattern T^q to RDF terms in T^D, such that applying the mapping by replacing each variable with its bound RDF term yields a subgraph T_q^D of T^D. We denote the set of solution mappings for BGP query as Ω and the set of partial solution mappings or bindings for a single triple pattern $t^q \in T^q$ as Ω_{t^q}.

A BGP query is evaluated by retrieving triples matching the patterns $t^q \in T^q$, and by performing a series of joins between the bindings Ω_{t_q} created from the triples retrieved. In particular, this is done for every two triple patterns that share a variable, forming a *join pattern* (that variable is referred to as the *join variable*).

In the Linked Data context, triple patterns are not evaluated on a single source, but have to be matched against the union of all sources D. When all sources in D are known, sources needed for processing a given query can thus be determined, retrieved by dereferencing their URIs, and triples obtained from these sources are joined along query join patterns [2]. In contrast to standard federated query processing triples from Linked Data can only be obtained via URI lookups, as opposed to retrieving only those matching a given pattern.

Exploration-based Linked Data Query Processing. In this work, we consider the case more general than in [2], following the direction of *exploration-based* Linked Data query processing, which does not rely on having complete knowledge about all Linked Data sources [3,6]. This line of approaches deal with the case where Linked Data is assumed to be dynamically evolving such that obtaining results might also require run-time discovery of new sources based on link traversal. In [3], no knowledge is available at all, and the query is assumed to contain at least one constant that is a URI. This URI is used for retrieving the first source representing the "entry point" to Linked Data; new sources are then discovered in a bottom-up fashion beginning from links found in that entry point. The approach recently proposed in [6] combines the two previous approaches [2,3] to discover new sources as well as leveraging known sources.

Here, query processing also relies on performing joins according to a query plan. The difference is that sources have to be discovered and even in the case where some knowledge is available, all source data are assumed to be remote and thus have to be retrieved from the network. Dealing with the network latency resulting from this requires that join operators do not block, such that when input is stalled for one part, progress can be made for other parts of the query plan. Two alternatives have been proposed to deal with this problem: one is *NBIJ* and the other is the *push-based SHJ*. Through a more systematic study of these operators we will show based on detailed cost models that the push-based SHJ is less expensive. Further, it also computes all results from retrieved data, while this completeness guarantee cannot be provided by NBIJ.

Remote and Local Linked Data Query Processing. While all approaches proposed so far assume remote data, in realistic scenarios, some Linked Data may be available locally. Conceptually, local data can be seen as yet another source. Thus, a *basic solution* to integrate locally stored data is to treat them just like a remote source and process them in the same way.

However, the availability of local data makes a great difference in practice, because while remote Linked Data sources have to be retrieved entirely (only URI lookup is available), local data can be accessed more efficiently using specialized indexes. Typically, local data are managed using a triple store, which maintains different indexes to directly retrieve triples that match a given pattern, i.e., relevant bindings Ω_{t^q} of a local source d can be directly obtained for $t^q \in T^Q$.

Given such querying capabilities for local data, we will show in this work that remote and local Linked Data with different access options can be processed using a single join operator. Instead of loading all local data, this operator retrieves only triples matching a given pattern. Further, we observe that there are non-discriminative triple patterns such as $\langle ?x, \mathit{rdf{:}type}, ?y\rangle$, which produce a large number of triples that do not contribute to the final results. To alleviate this problem, we take advantage of the available indexes to further instantiate query triple patterns with data obtained during query processing to load only triples that are guaranteed to produce join results.

3 Symmetric Index Hash Join

We propose to extend the SHJ operator to obtain the index-based SIHJ operator that can also take advantage of local data and indexes. This operation is similar to the index nested-loop join operator, where tuples (i.e., bindings) of one input are used to access indexes available for the other input. As opposed to that, it is a *non-blocking* operator based on SHJ, and is a hybrid one in that it employs both *push- and pull-based* mechanisms.

Query Processing based on SIHJ. Typically, a tree-shaped query plan is employed to determine the order of execution. Using the standard *SHJ* operator, the execution is pull-based in that starting from the root operator, higher-level operators in the plan invoke (the `next` method of) lower-level operators to obtain their inputs. Instead of pulling from the root, the *push-based SHJ* [6] allows inputs to be pushed to higher level operators (by invoking their `push` methods). The push-based SHJ maintains two hash tables, one for each input. Incoming tuples on either input are first inserted into their respective hash table and then used to probe the other hash table to find valid join combinations, which finally are pushed to subsequent operators. Thereby, results can be produced as soon as input tuples arrive. Without local data, SIHJ is essentially a push-based SHJ. Otherwise, it combines pull and push, i.e., while processing tuples that have been pushed to either one of its inputs, it also supports pulling local data from the index available for one of its inputs using data of the other input.

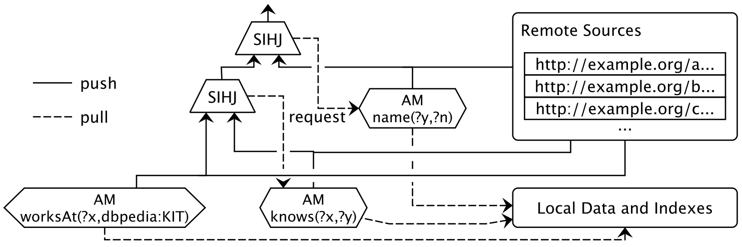


Fig. 1. Query plan with SIHJ operators and access modules (AM)

For a query with three triple patterns, Fig. 1 shows a left-deep query plan consisting of SIHJ operators and access modules for loading data. In a left-deep plan, the left input of all join operators is connected to the output of a join operator lower in the query plan, while the right input is connected to data sources, which in our case, might comprise both remote and local data. The exception is the lowest join operator, whose left input is not connected to another join operator but to data sources. Data arriving from remote sources are retrieved by a dedicated retrieval thread [6] and their data is pushed directly into the corresponding operators, whereas the access modules pull data from local indexes on request and then push them into the join operators.

Algorithm. In particular, we designate the left input of SIHJ as the "driving" input. All bindings that arrive on the left are used to perform lookups on local data to load only bindings into the right input that produce join results. This is achieved by instantiating the triple pattern on the right input with bindings for the join variable obtained from the left input:

Definition 3. *Let t_i^q, t_j^q be two triple patterns of T^q, v the join variable shared by t_i^q and t_j^q and $\Omega_{t_i^q}$ be the set of bindings for t_i^q. The results of the join of t_i^q and t_j^q on v is then calculated as $\Omega_{t_i^q} \bowtie_v \Omega_{t_j^q}$, where $\Omega_{t_j^q} = \bigcup_{u \in \Omega_{t_i^q}(v)} \{b | b \in \Omega_{t_j^q(v,u)}\}$, where $t_j^q(v,u)$ is an instantiated triple pattern obtained by substituting constant u for variable v.*

For local data we use separate *access modules* [11] that encapsulate access to local indexes. The `load` method for the AM is specified in Alg. 1. For every

SIHJ operator, one access module is created and connected to its right input. The access module accepts requests from the join operator *in* for loading the bindings Ω_{t^q} from triples matching a triple pattern t^q using the index I (line 1). All access to local storage is executed asynchronously by the access module so that operations in other parts of the query plan can still progress. Bindings loaded by an access module are pushed into its join operator (line 2).

Algorithm 1. AM: $load(in, t^q)$

```
   Input: Operator $in$, which requests data inputs for pattern $t^q$
1  $\Omega_{t^q} = I$.lookup($t^q$);                        // lookup in local index
2  foreach $b \in \Omega_{t^q}$ do  $in$.push($this$,$b$);     // push bindings to join operator
```

Algorithm 2. SHJ: $push(in, b)$

```
   Input: Operator $in$ from which input binding $b$ was pushed
   Data: Hash tables $H_i$ and $H_j$; current operator $this$; subsequent operator $out$;
         join variable $v$; $t^q_i$ is the left and $t^q_j$ the right triple pattern
1  if $in$ is left input then
2      if $b(v) \notin H_i$ then  $AM$.load($this$,$t^q_j(v, b(v))$)
3      $H_i[b(v)] \leftarrow H_i[b(v)] \cup b$
4      $J \leftarrow H_j[b(v)]$
5  else
6      $H_j[b(v)] \leftarrow H_j[b(v)] \cup b$
7      $J \leftarrow H_i[b(v)]$
8  forall $j \in J$ do  $out$.push($this$,merge($j$,$b$))
```

This use of local data via the access module is shown in Alg. 2. In particular, loading from the index results in a new triple that is pushed to the SIHJ operator (right input, line 4). All inputs of the "driving" left input are also pushed into this operator. When a binding b arrives on the left input, the corresponding hash table H_i is first probed to determine if it already contains the binding $b(v)$ for the join variable v captured by b (line 2). If this is not the case, i.e., this binding has not been processed before, a request to load triples from the local index using the instantiated triple pattern $t^q_j(v, u)$ is sent to the access module (line 2). Then, b is inserted into the corresponding hash table H_i and H_j of the right input is probed to obtain valid join combinations (line 3 - 4), which are then pushed to operator *out* (line 8). Bindings arriving on the right input (i.e., from remote sources or those pushed from the AM) are processed in a similar manner, except that no requests are sent to the access module (line 6 - 7), which is not necessary as all bindings are stored in hash table H_j and are therefore available when a matching input arrives on the left input.

Note that bindings on the right or left input may be both local or remote data. Both remote and local data may be pushed into the left input. Remote data may also be pushed into the right input, and through explicit pulling using the AM (line 4), this input might also contain local data.

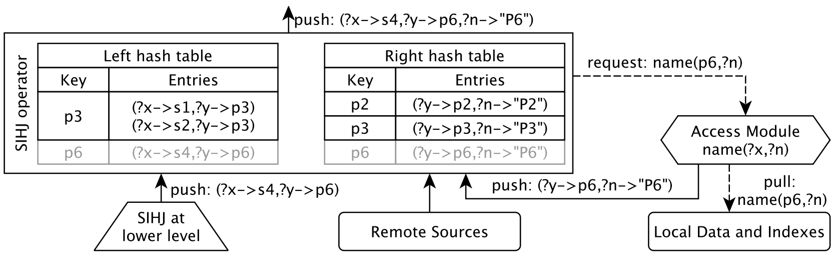


Fig. 2. Processing of the SIHJ operator for data coming from the left input

Fig. 2 illustrates the operation of a SIHJ operator. An input containing bindings for two variables $?x, ?y$ is received and then inserted into the left hash table. Then a request for $\langle p6, name, ?n\rangle$ is sent to the access module. After loading the data from the local index, the binding for $?y, ?n$ is inserted into the right hash table. In combination with the binding in the left hash table a join result is finally created and pushed to the subsequent operator.

Cost Model. We use a unit-time-basis cost model that captures the operator cost in terms of the tuples that are accessed and the cost of the physical operations needed [4]. All costs are defined in an abstract manner, independent from the concrete implementation and data structures being used.

The cost of a SIHJ with two inputs A and B is the sum of three components: the cost for joining tuples arriving on the left and the right input and the cost of the access module: $C_{A \bowtie B} = C_{A \rtimes B} + C_{A \ltimes B} + C_{AM}$

The operation carried out for tuples on the left input are: insertion into hash table for A, probing of hash table for B, creating join results and finally, sending a request to the access module. Accordingly, the cost $C_{A \rtimes B}$ is defined as follows:

$$C_{A \rtimes B} = |A|(I_h + P_h + \varphi \cdot |B| \cdot J \cdot \frac{|A|}{|A| + |B|} + R)$$

with: weight factors I_h, P_h for hash table insert and probe; join selectivityφ; weight factor J for creating result tuples; weight factor R for request to access module; the fraction $\frac{|A|}{|A|+|B|}$ of inputs arriving on the left input

The term $I_h + P_h$ represents the cost of inserting an incoming tuple and then probing the other hash table. Given a join selectivity φ, the number of results for $A \bowtie B$ is $\varphi|A||B|$. Multiplied by the weight factor for creating results, this yields the term $J \cdot \varphi \cdot |A||B|$. Further, it is multiplied with $\frac{|A|}{|A|+|B|}$ to consider join cost only for tuples that actually arrive in A. For each tuple in A, a request is sent to the access module, whose cost is captured by R.

The cost $C_{A \ltimes B}$ for the other input is defined in a similar fashion, except that no requests to the access module are needed:

$$C_{A \ltimes B} = |B|(I_h + P_h + J \cdot |A| \cdot \varphi \cdot \frac{|B|}{|A| + |B|})$$

The cost C_{AM} for the access module is defined as $C_{AM} = |A| \cdot P_l + |B_l| \cdot L_l$, where the input B is split into tuples from remote sources B_r and local tuples loaded from disk B_l (i.e., $B = B_r \cup B_l$ and $B_r \cap B_l = \emptyset$).The cost for probing the local index, which has to be done for all tuples arriving in A, is represented by $|A| \cdot P_l$. When matching tuples are found, they have to be loaded from disk, the cost of which is given by $|B_l| \cdot L_l$.

Using the Cost Model for Query Optimization. The cost model developed in the previous section abstracts from concrete implementations and hardware by using weight factors. To use the cost model for query optimization these weight factors have to be known. The weight factors can be determined by running the operator on known input and then measuring the CPU time of the operations represented by the individual weight factors. Note that the weight factors are dependent on the characteristics of the data being used, in particular on the input size (both remote and local) and join selectivity. For example, the higher the join selectivity, the higher the relative weight of join result creation. Thus – as always the case of query optimization in practice – weight factors shall be derived from the underlying data.

In particular, measurements shall be taken for different combinations of input size and join selectivity. These measurements shall aim at covering a large space of possible combinations. At query compile-time, the weight factors precomputed for the combination that best fit the input size and join selectivity estimated for the given query are used to estimate join operator cost.

Batching. When an access module receives a request for loading data matching an instantiated triple pattern from local storage, all matching triples will have the same binding for the join variable because it has been used to instantiate the triple pattern in the first place. Sending each binding one by one to the join operator will incur an unnecessary overhead because they all will be inserted into the same hash bucket; and subsequently, the same hash bucket has to be probed several times when using these bindings. It is therefore beneficial to process data loaded from local indexes in *batches*, where the hash tables of the join operator are accessed only once for a batch of bindings.

4 Comparison to Non-blocking Iterator

In [3], NBIJ was proposed to deal with high network latency in the Linked Data context and the resulting issue of blocking. We now study this operator, extending previous work [3] with a completeness analysis and cost model.

Query Processing based on NBIJ. NBIJ is based on a traditional pull-based mechanism, i.e., each operator in the query plan has a `next` method that is called by operators higher in the query plan tree. It is also used in left-deep plans, where all inputs consist only of data from remote sources.

During query processing an in-memory list G of data sources is maintained. Each downloaded source is indexed and then added separately to G. When the next method receives a result from a lower operator on the left input, first the following requirement is checked:

Requirement 1. *Let t_i^q, t_j^q be two triple patterns of T^q, v the join variable shared by t_i^q and t_j^q and $b \in \Omega_{t_i^q}(v)$ a binding received on the left input.*

Then b can only be further processed if the following condition holds: $\forall u \in \{s(t^q_j(v, b(v))), p(t^q_j(v, b(v))), o(t^q_j(v, b(v)))\}$: *if u is an URI then $ID(u) \in G$.*

This requirement ensures that all sources identified by URIs in the instantiated triple pattern have been retrieved and added to the list of in-memory sources. If the requirement is not fulfilled, the sources are marked for asynchronous retrieval, the binding is *rejected* by calling the `reject` method of the lower join operator, and the operator calls `next` again to retrieve further inputs. Otherwise, all sources in G are successively queried for the instantiated triple pattern $t^q_j(v, b(v))$ using in-memory indexes to construct join results.

When the `reject` method of a NBIJ operator is called, the rejected binding is added to a separate list maintained by the operator. On subsequent calls to its `next` method, the operator randomly decides between returning a previously rejected binding from the list or a new one. The rejection mechanism ensures that query processing can proceed even when sources for a particular pattern are not yet available.

Completeness. A disadvantage of NBIJ is that the obtained results are not necessarily complete w.r.t. downloaded data, i.e., it is not guaranteed that all possible results that can be derived from downloaded data are actually computed [3]. While Requirement 1 does ensure that all sources mentioned in an instantiated triple pattern are retrieved before processing the pattern, it is possible that data matching that pattern is contained in other sources *retrieved later* during query processing. This is possible because Linked Data sources can contain arbitrary data and therefore not all data matching a particular triple pattern is necessarily contained in the sources mentioned in the pattern. As the NBIJ works in a pull-based fashion (and not push-based), this data will be disregarded if it is never requested again.

In contrast, a query plan based on SIHJ operators is guaranteed to produce all results. Requirement 1 is not necessary, because the operation of the SIHJ operator is completely symmetrical and push-based, i.e., incoming data can arrive on both inputs and in any order and its operation is driven by the incoming data instead of the final results. When an input tuple arrives on either of its input, the SIHJ operator is able to produce all join results of that tuple with *all previously seen inputs*, because these are kept track of in the hash table of the SIHJ operator. This ensures that it does not matter at which point during query processing a particular input for a triple pattern arrives, the final result is always complete with respect to the data in the sources that were retrieved.

Cost Model. Since the randomness of the rejection mechanism cannot be accurately captured in a cost model, we simply assume that all incoming bindings on the left input are first rejected and then processed on the second try. The cost for the NBIJ operator can then be calculated as follows:

$$C_{A \bowtie_{NBIJ} B} = |A|(P_G + T + |G| \cdot L) + \varphi|A||B| \cdot J$$

with: weight factor P_G for checking Req. 1; number of sources $|G|$; weight L for probing in-memory graph; weight T for tracking rejected bindings

The term P_G gives the cost for checking whether the corresponding sources for a binding have been retrieved. The cost for rejecting a binding is T. Both these operations are performed for all bindings of the left input. For each bindinge from A all available graphs (in the worst case all sources) are consecutively probed for join combinations, yielding the term $|A||G| \cdot L$.

We now compare the cost models of the SIHJ and NBIJ operators. As the NBIJ operator only operates on remote data, we disregard the costs of SIHJ for requests sent to the access module. The cost of SIHJ is then:

$$C_{A \bowtie_{SIHJ} B} = |A|(I_h + P_h) + |B|(I_h + P_h) + \varphi|A||B| \cdot J$$

Assuming that both operators operate on the same inputs (i.e., we disregard the completeness issue discussed earlier), the results produced by both operators are the same and therefore the cost $\varphi|A||B| \cdot J$ for creating results is the same. The SIHJ might incur higher overhead for maintenance of its hash tables as all incoming tuples require insertion into and probing of a hash table. Compared to this, NBIJ incurs cost for checking the requirement, rejecting bindings and maintaining rejected bindings. However, NBIJ further incurs cost for probing all in-memory sources $|A||G| \cdot L$, which depends on the number of available sources. That means that the more sources are retrieved during processing, the higher the cost of the operator, whereas the SIHJ operator incurs no such cost and is independent from the number of retrieved sources.

5 Related Work

Previous work on Linked Data query processing [3,2,6] was discussed throughout the paper. Here, we discuss related database research.

Join Operators. In the database community a lot of research has been done on join operators that can produce results as soon as inputs become available without blocking and are therefore suited to high latency environments and stream processing. The symmetric hash join [15] was the first of a new generation of such operators. To deal with the high memory requirements of the SHJ, the XJoin operator [13] flush tuples to disk if memory becomes scarce (during the arriving phase). During a reactive phase, when inputs are blocked, XJoin uses previously flushed tuples to produce further join results. During the final cleanup phase after all inputs have been consumed, the XJoin operator joins the remaining tuples that were missed during the previous phases. An important observation is that the output rate is heavily influenced by which tuples are flushed to disk, as some tuples might produce more results than others. This lead to the introduction and subsequent improvement of a flushing policy [9,12,1].

The SIHJ operator proposed in this work is also based on the symmetric hash join. The memory consumption of the SIHJ could be addressed using concepts proposed for the XJoin; but this topic was not the focus of this work. Similar to XJoin, SIHJ does access locally stored data, but the purpose is different: SIHJ treats local data as an additional data source whereas XJoin and the mentioned work based on it use the disk as a cache and focus on the problem of how to use it for tuple storage when memory becomes scarce.

Adaptive Query Processing. Access Modules [11] were proposed to be used in conjunction with an Eddy to provide different data access methods (scan, index) and switch between them at run-time. Probe tuples are sent from the Eddy to the access module to request a particular subset of the data. The access module then pushes the data into the Eddy, marking the end with a special tuple. In our work we adopt the notion of an Access Module to provide access to local indexes in an asynchronous fashion.

Stream Databases. Fjords [8] support push- and pull-based operators and combine push-based stream processing with pull-based processing. Fjords provide a bounded queue between operators that buffers tuples between two operators so that push- and pull-based operators can be used in the same query plan. Because the queues are bounded, tuples may have to be discarded. The SIHJ operator also uses push- and pull- based processing, but in a single operator.

In all, some concepts underlying SIHJ overlap with ideas from related database work. However, there is no single operator that can be used for remote and local data where the latter is not considered as cache but an additional independent source – especially in the Linked Data setting. SIHJ fills this gap and presents a means to incorporate local data into Linked Data query processing.

6 Evaluation

The evaluation consists of two parts: first, we use real-world datasets to compare SIHJ with NBIJ; second, we create several synthetic datasets with different characteristics to study the performance based on the proposed cost models. We present a summary and refer to the technical report [7] for more details.

6.1 Overall Performance

Setting. In this part, we first show the benefits of stream-based query processing in comparison to non-blocking iterators. We compare an SHJ-based implementation (*SQ*) with the implementation of the NBIJ-based query processing (*NBI*) in SQUIN[1]. Both systems do not use local data and run without query optimization, and thus are comparable. Second, we compare three implementations of stream-based query processing over local and remote data to study the push- and pull-based mechanism. One is the baseline, which is a configuration of SIHJ that does not pull from the local data indexes but simply pushes all relevant data into the query plan (*SQ-L*), i.e., this corresponds to the basic solution described in Section 2. This is compared with the configuration using indexes as proposed in this work, where *SQ-I* ran without and *SQ-IB* ran with batching.

All experiments were run on a server with two Intel Xeon 2.8GHz Dual-Core CPUs and 8GB of main memory. SQUIN is a Java implementation of NBIJ, whereas the SQ systems are implemented in Scala. Both systems employ multithreading and were configured to use five threads to retrieve sources.

Dataset. The data consists of several popular Linked Data datasets, among them DBpedia, Geonames, New York Times, Semantic Web Dog Food and several life

[1] `http://www.squin.org`

science datasets. In total, the data consists of ca. 166 million triples. For the experiments with approaches using local data, the dataset was split into remote and local data, where the randomly chosen local data accounted for 10% of the total dataset. Remote data were deployed on a CumulusRDF[2] Linked Data server on the local network so that data can be accessed using URI lookup, whereas local data were indexed using our triple store [14].

Queries. We created 10 BGP queries that cover different complexities w.r.t. query size and the number of sources retrieved during query processing. For example, Q1 retrieves the names of authors of demo papers at ISWC 2008:

```
SELECT * WHERE { ?p sw:isPartOf <http://data.semanticweb.org
      /conference/iswc/2008/poster_demo_proceedings> .
  ?p swrc:author ?a .  ?a rdfs:label ?n . }
```

Results. Fig. 3a shows query times of the SQ and NBI systems for all ten queries. The SIHJ-based system was faster for all queries, in some cases up to an order of magnitude. On average, queries took 9699.18ms for SQ and 41704.27ms for NBI, corresponding to an improvement of 77%.

Query times for SQ-I, SQ-IB and SQ-L are presented in Fig 3b. In all cases, SQ-I and SQ-IB outperformed SQ-L and also here, improvements were up to an order of magnitude in some cases. Note that for Q8, SQ-L ran out of memory because the amount of local data to be loaded was too large. On average, query times were 9366.39ms for SQ-IB, 9396.18ms for SQ-I and 28448.7.98ms for SQ-L. This yielded an improvement of 67% of SQ-IB over SQ-L, clearly showing that using locally available indexes is beneficial. It reduced the amount of data that is loaded from disk, especially for queries with less selective triple patterns. The improvement achieved through batching could also be observed, and will be examined in more detail in the next section.

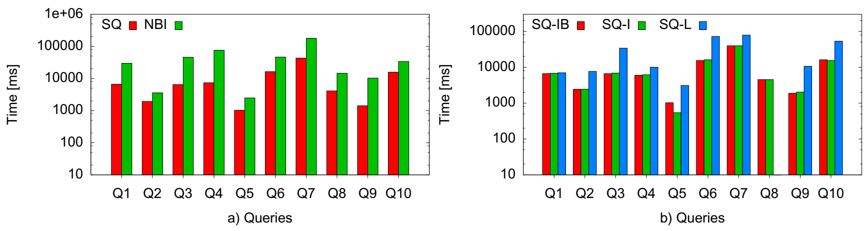


Fig. 3. Overall query times for a) SQ and NBI and b) SQ-IB, SQ-I and SQ-L

6.2 Join Operator Performance in Detail

Setting. Previously, the operators were incorporated into plans for processing entire BGP queries. Here, we focus on *join processing* using SIHJ and NBIJ. Synthetic datasets that have known characteristics are used to examine the performance of these operators in detail. We evaluated three SIHJ-approaches:

[2] `http://code.google.com/p/cumulusrdf/`

SQ-IB, SQ-I and SQ. For the NBIJ operator, we used our own implementation in order to instrument the code with detailed measurement points.

Datasets. The synthetic datasets for these experiments consist of separate sets of triples for the left and right input. The right input is split into local and remote parts, where the remote part is distributed among a number of sources. Here, we want to focus on the weight factors of the cost models and therefore keep "remote" data in memory and push it into the operator, instead of performing network access, which might lead to inconsistencies in the performance measurements. The data were generated with the following parameters: the size of the left and right input is given by a, b, respectively; ρ is the fraction of the right input that is local data; φ is the join selectivity; the number of sources for the remote part of the right input is s (the source sizes follow a normal distribution). We create several sets of datasets, where one of the parameters is changed while the others are fixed in order to examine the influence of each parameter.

Results. We examine the parameters' effect on the weights of the cost model:

Join Selectivity. Fig. 4a shows the influence of join selectivity on the different weight factors of the SIHJ cost model in terms of their relative fraction of total time measured for SQ. For joins with high selectivity ($\varphi = 0.0005$), i.e., only a small number of input tuples match other tuples to form join results, loading of local data took the largest part of total processing time. For low selectivity joins ($\varphi = 0.5$), the creation of result tuples dominated query processing. Using the cost model, this can be explained by the observation that join selectivity has impact only on the term $J \cdot \varphi |A||B|$, meaning that only the weight of result creation increases with lower join selectivities.

Number of Sources. Fig. 4b shows processing times for various number of sources $|G|$ for SQ and NBI. Overall, times for SQ were largely the same for all source counts, whereas the times for NBI increased with larger a number of sources. The times for NBI were split into times for checking the requirement (cr), loading data from the in-memory graphs (load) and creating result tuples (join). Clearly, results show that both cr and load times were dependent on $|G|$. This is accounted for by the cost model, i.e., the term $|A|(P_G + |G| \cdot L)$ indicates that cost depends on P_G and $|G|$. Join times were the same because the number of results does not change with $|G|$.

Input Size. Fig. 4c presents the effect of input size (on the right input) on processing times of SQ-I and SQ-IB. We can see that for larger inputs, the relative time spent on loading local data decreased and the relative weights of hash table insertion and increased. This is probably due to the larger hash tables that were required for larger input sizes, introducing more overhead for rehashing when the hash tables need to be expanded.

Local Data Fraction. We examined processing times for various local data sizes. The overall number of inputs on the right input was the same, only the ratio between remote and local data changed. A value of $\rho = 0.1$ means 10% of the data is local data. Fig. 4d shows processing times for SQ-I and SQ-IB. With higher local data fractions, the impact of loading on total processing times is more pronounced. Whereas for a local fraction of 0.1 loading accounted for about 22% of total time, at 0.8 it accounted for over 60%. This is because with more local data, more effort was spent on using the local indexes to find triples that produce join results. Thus, less effort was needed for join, probe as well as insert. Note that as remote data were actually in-memory data, access to local data was slower than for "remote" data. Thus, loading here essentially means loading local data. In the standard setting, network access is usually slower than disk access. This means that loading would have an even larger impact.

This experiment also shows the benefits of batching, which are more pronounced for larger amounts of local data, as reflected by the smaller amounts of time spent on inserting and probing hash tables.

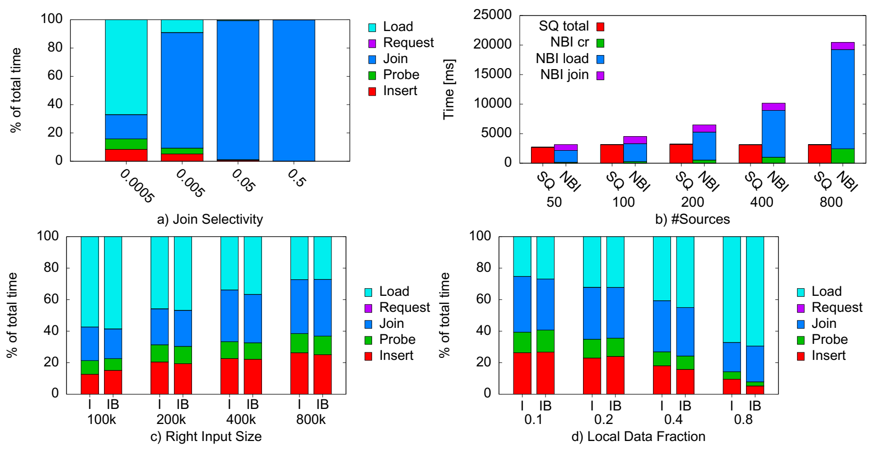


Fig. 4. a) Join selectivity ($b = 10000, \rho = 0.2, s = 200$), b) Number of sources ($b = 500000, \varphi = 0.0002, \rho = 0.2$), c) Input size ($\varphi = 0.2, \rho = 0.2, s = 200$), d) Local part ($b = 500000, \varphi = 0.0002, s = 200$) ($a = 10000$ for all)

7 Conclusion

We propose a new operator, the Symmetric Index Hash Join (SIHJ) for processing queries over local and remote Linked Data in a stream-based and non-blocking fashion. We provide cost models for SIHJ and the Non-Blocking Iterator (NBIJ) previously proposed for dealing with remote Linked Data. A detailed comparison shows that while SIHJ might have larger overhead for accessing its hash tables, its cost does not depend on the number of data sources processed. The number of sources however has a large impact on the performance of NBIJ. Further, as opposed to NBIJ, SIHJ guarantees complete results w.r.t. the data retrieved during query processing. We performed an evaluation of both operators on a real-world dataset and several synthetic datasets. We show that stream-based query processing using push-based SHJ performs on average 77% better than NBIJ-based query processing w.r.t. to overall query execution time. The experiments show that using available indexes to access local data is beneficial, resulting in an average improvement of 67% compared to a baseline that simply loads all data matching query triple patterns. Detailed analyses using the synthetic datasets further shed light on the weights of the proposed cost models.

Acknowledgements. The authors would like to thank Olaf Hartig for providing an updated implementation of SQUIN and Andreas Harth for discussions. Research reported in this paper was supported by the German Federal Ministry of Education and Research (BMBF) in the CollabCloud project (01IS0937A-E).

References

1. Bornea, M., Vassalos, V., Kotidis, Y., Deligiannakis, A.: Double index NEsted-Loop reactive join for result rate optimization. In: IEEE 25th International Conference on Data Engineering, ICDE 2009, pp. 481–492 (2009)
2. Harth, A., Hose, K., Karnstedt, M., Polleres, A., Sattler, K., 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, New York (2010)
3. 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)
4. Jeffrey, J.K., Naughton, J.F., Viglas, S.D.: Evaluating window joins over unbounded streams. In: ICDE, pp. 341—352 (2003)
5. Klyne, G., Carroll, J.J., McBride, B.: Resource description framework (RDF): concepts and abstract syntax (2004)
6. 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)
7. Ladwig, G., Tran, T.: SIHJoin: Querying Remote and Local Linked Data. Technical report (2010), http://people.aifb.kit.edu/gla/tr/sq_report.pdf
8. Madden, S., Franklin, M.J.: Fjording the stream: An architecture for queries over streaming sensor data. In: International Conference on Data Engineering, p. 0555. IEEE Computer Society, Los Alamitos (2002)
9. Mokbel, M., Lu, M., Aref, W.: Hash-merge join: a non-blocking join algorithm for producing fast and early join results. In: Proceedings of 20th International Conference on Data Engineering, pp. 251–262 (2004)
10. Prud'hommeaux, E., Seaborne, A.: SPARQL Query Language for RDF. W3C Recommendation (2008)
11. Raman, V., Deshpande, A., Hellerstein, J.: Using state modules for adaptive query processing. In: Proceedings of 19th International Conference on Data Engineering, pp. 353–364 (2003)
12. Tao, Y., Yiu, M.L., Papadias, D., Hadjieleftheriou, M., Mamoulis, N.: Rpj: Producing fast join results on streams through rate-based optimization. In: SIGMOD Conference, pp. 371–382 (2005)
13. University, T.U., Urhan, T., Franklin, M.J.: XJoin: a Reactively-Scheduled Pipelined Join Operator (2000)
14. Wang, H., Liu, Q., Penin, T., Fu, L., Zhang, L., Tran, T., Yu, Y., Pan, Y.: Semplore: A scalable ir approach to search the web of data. J. Web Sem. 7(3), 177–188 (2009)
15. Wilschut, A.N., Apers, P.M.G.: Dataflow query execution in a parallel main-memory environment. Distributed and Parallel Databases 1(1), 103–128 (1993)

Zero-Knowledge Query Planning for an Iterator Implementation of Link Traversal Based Query Execution

Olaf Hartig

Humboldt-Universität zu Berlin
hartig@informatik.hu-berlin.de

Abstract. Link traversal based query execution is a new query execution paradigm for the Web of Data. This approach allows the execution engine to discover potentially relevant data during the query execution and, thus, enables users to tap the full potential of the Web. In earlier work we propose to implement the idea of link traversal based query execution using a synchronous pipeline of iterators. While this idea allows for an easy and efficient implementation, it introduces restrictions that cause less comprehensive result sets. In this paper we address this limitation. We analyze the restrictions and discuss how the evaluation order of a query may affect result set size and query execution costs. To identify a suitable order, we propose a heuristic for our scenario where no a-priory information about relevant data sources is present. We evaluate this heuristic by executing real-world queries over the Web of Data.

1 Introduction

While the possibility to query the emerging Web of Data enables exciting opportunities, executing SPARQL queries over the Web poses novel challenges [1]. It is impossible to know all data sources that might contribute to the answer of a query. To tap the full potential of the Web, traditional query execution paradigms are insufficient because they assume knowledge of a fixed set of potentially relevant data sources beforehand. In [2] we propose a novel query execution paradigm that conceives the Web of Data as an initially unknown set of data sources and makes use of the characteristics of Linked Data, in particular, the existence of links between data items from different sources. The main idea of our approach is to intertwine the construction of the query result with the traversal of data links that correspond to intermediate solutions in the construction process. This strategy, which we call *link traversal based query execution*, allows the execution engine to discover potentially relevant data during the query execution.

Different implementations of the general idea of link traversal based query execution are possible (e.g. [2,3]), each having its own strengths and drawbacks. In [2] we propose an iterator based implementation approach, including concepts that improve its execution times. This implementation approach applies a synchronized pipeline of operators that evaluate the query in a fixed order. While

© Springer-Verlag Berlin Heidelberg 2011

this approach determines query results efficiently, the fixed evaluation order may prevent finding some query results. In this paper we address this limitation.

As a prerequisite to analyze the iterator based approach we provide a definition of link traversal based query execution, independent of possible implementation approaches, and, thereby, introduce a completeness criteria. We align our iterator based approach with this definition: We prove that the approach is sound and analyze why it cannot guarantee results that satisfy our completeness criteria. Furthermore, we describe how the evaluation order of the query may affect result completeness. Since this effect causes the need to select a suitable order, we discuss the possibilities of query planning and propose a heuristics based approach as the only applicable strategy in our scenario in which we cannot assume any information about statistics or data distribution when we start the execution of a query. To evaluate our heuristic we execute real-world queries.

This paper is structured as follows: While Section 2 defines link traversal based query execution, Section 3 aligns our implementation approach with this definition and analyzes the issue of result completeness. Section 4 discusses query planning and our heuristic for plan selection. In Section 5 we evaluate this heuristic. Finally, we study related work in Section 6 and conclude in Section 7.

2 Definition of Link Traversal Based Query Execution

Link traversal based query execution is a new query execution paradigm developed to exploit the Web of Data to its full potential. Since adhering to the Linked Data principles is the minimal requirement for publication in the Web of Data our approach relies solely on these principles instead of assuming the existence of source-specific query services such as SPARQL endpoints. This section provides a formal definition of the general idea of link traversal based query execution. For the formalization we adopt a static view of the Web, that is, we assume no changes are made to the data on the Web during the execution of a query.

2.1 Preliminaries

The Linked Data principles require to describe data using RDF. RDF distinguishes three distinct sets of *RDF terms*: U, the (possibly infinite) set of URIs, L, an infinite set of literals, and B, an infinite set of blank nodes that represent unnamed entities. An *RDF triple* is a 3-tuple $t = (s, p, o) \in (U \cup B) \times U \times (U \cup B \cup L)$ where s is called the *subject* of t, p the *predicate*, and o the *object*.

In the Web of Data entities have to be identified via HTTP scheme based URIs. Let $U^{\mathrm{LD}} \subset U$ be the (possibly infinite) set of all these URIs. By looking up such a URI we retrieve RDF data about the entity identified by the URI. For our formalization we introduce a function, denoted as *lookup*, to refer to the result of such look-ups: *lookup* is a surjective function which returns for each URI $u \in U^{\mathrm{LD}}$ a *descriptor object*, that is, a set of RDF triples which i) can be retrieved by looking up u on the Web and which ii) describes the entity identified by u. Hence, based on the Linked Data principles we expect:

$\forall u \in U^{\mathrm{LD}} : \big(\exists (s,p,o) \in \mathit{lookup}(u) : s = u \vee o = u\big)$. Note, *lookup* is not injective; it is possible that the same descriptor object is retrieved by looking up distinct URIs. In this case, the descriptor object describes multiple entities. For each $t \notin U^{\mathrm{LD}}$ the look-up function returns an empty descriptor object: $\mathit{lookup}(t) = \varnothing$.

We define our query execution approach for basic graph patterns[1]. A *basic graph pattern* (BGP) is a subset of the set[2] $(U \cup V) \times (U \cup V) \times (U \cup V \cup L)$ where V is an infinite set of query variables. The elements of a BGP are called *triple patterns*. For each triple pattern tp we write $\mathit{uris}(tp)$ and $\mathit{vars}(tp)$ to denote the set of all URIs and the set of all query variables contained in tp, respectively. A *matching triple* in a set of RDF triples G for a triple pattern $(\tilde{s}, \tilde{p}, \tilde{o})$ is any RDF triple $(s,p,o) \in G$ with $(\tilde{s} \notin V \Rightarrow \tilde{s} = s) \wedge (\tilde{p} \notin V \Rightarrow \tilde{p} = p) \wedge (\tilde{o} \notin V \Rightarrow \tilde{o} = o)$.

2.2 Link Traversal Based Solutions for Basic Graph Patterns

We define the notion of solutions for the link traversal based query execution of a BGP using a two-phase approach: First, we define what descriptor objects can be discovered during link traversal based query execution. Then, we formalize solutions as sets of variable bindings that correspond to a subset of all data from all discovered descriptor objects. Notice, while this two-phase approach provides for a straightforward definition of solutions it does not correspond to the actual query execution strategy of intertwining the traversal of data links and graph pattern matching as is characteristic for link traversal based query execution.

To formalize what descriptor objects can be discovered during the link traversal based execution of a BGP we introduce the concept of *reachability*:

Definition 1. *Let $b = \{tp_1, \dots, tp_n\}$ be a BGP; let D be a descriptor object. D is* reachable *by the execution of b iff either*

- $\exists (\tilde{s}, \tilde{p}, \tilde{o}) \in b : \mathit{lookup}(\tilde{s}) = D \vee \mathit{lookup}(\tilde{p}) = D \vee \mathit{lookup}(\tilde{o}) = D$
- *or there exists another descriptor object D', a triple pattern $tp \in b$, and an RDF triple $t = (s,p,o)$ such that i) D' is reachable by the execution of b, ii) t is a matching triple for tp in D', and iii) $\mathit{lookup}(s) = D$, $\mathit{lookup}(p) = D$ or $\mathit{lookup}(o) = D$.*

To represent the solutions of BGPs we adopt the notion of a *solution mapping* as defined in the SPARQL specification [4]. These mappings bind query variables to RDF terms. Hence, a solution mapping μ is a set of variable-term-pairs where no two pairs contain the same variable. The application of a solution mapping μ to a triple pattern tp, denoted as $\mu[tp]$, implies replacing each variable in tp by the RDF term it is bound to in μ; unbound variables must not be replaced. Similarly, a solution mapping μ can be applied to a whole BGP b: $\mu[b] = \{\mu[tp_i] \mid tp_i \in b\}$. Using solution mappings we introduce our notion of *solutions* for a BGP:

[1] While we consider only BGPs in this paper, the solutions for BGPs that might be determined using link traversal based query execution, can be processed by the SPARQL algebra that provides operators for more complex SPARQL graph patterns.

[2] For the sake of a more straightforward formalization we do not permit blank nodes in BGPs as is possible according to the SPARQL specification [4]. In practice, each blank node in a SPARQL query can be replaced by a new variable.

Definition 2. *Let b be a BGP and let $\mathcal{D}$ be the set of all descriptor objects reachable by the execution of b. A solution mapping μ is a* solution *for b iff i) it holds*[3]*: $\mu[b] \subseteq \bigcup_{D \in \mathcal{D}} D$ and ii) μ maps only these variables that are in b, i.e. $\forall (v,t) \in \mu : v \in \bigcup_{tp \in b} vars(tp)$,*

2.3 Link Traversal Based Construction of Solutions

While the two-phase definition approach in the previous section defines the notion of solutions for BGPs in the context of link traversal based query execution, it does not reflect the fundamental idea of intertwining link traversal with the construction of solutions. Instead, a query execution engine that would directly implement this two-phase approach would have to retrieve all reachable descriptor objects before it could generate solutions for a BGP. Hence, the first solutions could only be generated after all data links that correspond to triple patterns in the BGP have been followed recursively. Retrieving the complete set of reachable data can take a long time and may exceed the resources of the execution engine.

For this reason, the link traversal based query execution approach requires to construct the solutions incrementally, using a query-local dataset that is continuously augmented with additional descriptor objects. These descriptor objects are discovered by looking up URIs that occur in *intermediate solution*, that are, solution mappings from which the solutions are constructed. In the next section we discuss a possible implementation of this strategy.

3 Iterator Based Implementation

In [2] we introduce the idea of link traversal based query execution using an iterator based implementation of this idea. In this section we align this implementation approach with the general idea defined in the previous section. We give an introduction to the approach and discuss soundness and completeness.

3.1 Introduction to the Approach

Our implementation approach applies a synchronized pipeline of operators that evaluate a BGP in a fixed order. This pipeline is implemented as a chain of iterators $I_1, \ldots, I_n$ where each iterator I_i is responsible for triple pattern tp_i from the *ordered BGP*[4] $\bar{b} = [tp_1, \ldots, tp_n]$. The operation implemented by these iterators returns solution mappings that are solutions for a BGP consisting of the triple pattern of the corresponding iterator and all triple patterns of the

[3] For the union of descriptor objects we assume that no two descriptor objects share the same blank nodes. This requirement can be guaranteed by using a unique set of blank nodes identifiers for each descriptor object retrieved from the Web.

[4] We represent an ordered BGP as a list, denoted by comma-separated elements enclosed in brackets. In the remainder of this paper we conceive such an ordered BGP as a logical query plan. Selecting an order for a BGP is a query optimization problem as we discuss in Section 4. However, in this section we assume a given order.

Algorithm 1. `GetNext` function for our iterator based implementation approach.

Require: A set $\mathcal{D}$ of descriptor objects that always represents the current state of the query-local dataset; a triple pattern tp_i; a predecessor iterator I_{i-1} that provides solutions for $\{tp_1, \dots, tp_{i-1}\}$ in $\mathcal{D}$; an initially empty set M_i that allows the iterator to keep matching triples between calls to this iterator function

```
1: while $M_i = \varnothing$ do
2:    $\mu' := I_{i-1}$.GetNext
3:    if $\mu' =$ NotFound then return NotFound end if
4:    $tp'_i := \mu'[tp_i]$
5:    for all $u \in uris(tp'_i)$ do
6:       if $lookup(u) \notin \mathcal{D}$ then $\mathcal{D} := \mathcal{D} \cup \{lookup(u)\}$ end if
7:    end for
8:    $M_i :=$ set of all matching triples for $tp'_i$ in $\bigcup_{D \in \mathcal{D}} D$
9: end while
10: $t_j :=$ an element in $M_i$
11: $M_i := M_i \setminus \{t_j\}$
12: $\mu_j :=$ a solution mapping such that $\mu_j[tp'_i] = t_j$ and $\forall (v,t) \in \mu_j : v \in vars(tp'_i)$
13: return $\mu' \cup \mu_j$
```

preceding iterators; i.e., each I_i provides solutions for $\{tp_1, \dots, tp_i\}$. To determine these solutions each iterator executes the following three steps repetitively: First, the iterator consumes a solution mapping μ' of its direct predecessor[5] and applies this mapping to its triple pattern tp_i, resulting in a triple pattern $tp'_i = \mu'[tp_i]$ (lines 2 to 4 in Algorithm 1); second, the iterator ensures that the query-local dataset contains these descriptor objects that can be retrieved from looking up all URIs in tp'_i (lines 5 to 7); and, third, the iterator tries to generate solutions by finding matching triples for tp'_i in the query-local dataset (lines 8 to 13).

This approach is sound because each solution determined by the approach satisfies Definition 2 as we prove in [5]. Moreover, the approach is in fact an implementation of the idea of link traversal based query execution because it satisfies the criteria specified in Section 2.3: First, due to the second step, the iterators continuously augment the query-local dataset by looking up URIs on the Web and, second, all solutions are constructed incrementally, using this dataset.

The practicability of this approach is based on the following *look-up assumption*: If a solution mapping binds query variable v to URI u then the descriptor object $lookup(u)$ may contain matching triples for triple patterns that contain v. This assumption is justified by the common practice of publishing Linked Data.

3.2 Missing Query Results

Even if our iterator based approach is a correct implementation of link traversal based query execution, it does not guarantee to return all solutions that satisfy Definition 2. In the following we discuss the reasons for this limitation.

[5] We assume the first iterator, I_1, consumes a single, empty solution mapping once.

The most restricting characteristic of the iterator based approach is the fixed order in which it evaluates the triple patterns from a BGP. Since the discovery of reachable descriptor objects is aligned with the fixed-order evaluation of triple patterns, the approach cannot make use of the flexibility in the discovery as would be possible according to Definition 1. Hence, it may not discover all reachable descriptor objects and, thus, may miss some matching triples.

Additionally, the iterators dismiss an intermediate solution μ' consumed from their predecessor when they consume the next μ'. Hence, each μ' is used only once to find matching triples M_i for $\mu'[tp_i]$. Due to this "use and forget" strategy the approach misses matching triples for $\mu'[tp_i]$ that occur in these descriptor objects that will be discovered after the next μ' has been requested.

Finally, to enable an implementation that avoids inconsistencies and concurrency issues, the iterators determine all matching triples in isolation as represented by the set M_i in Algorithm 1. We propose to implement this strategy with an immutable snapshot of the query-local dataset [2,6]. However, by isolating the triple pattern matching, an iterator misses matching triples for $\mu'[tp_i]$ if they occur in descriptor objects that subsequent iterators discover when they consume the intermediate solutions generated from the current M_i, although the current μ' would still be available (in contrast to the aforementioned case).

Even if the iterator based approach may not return all solutions that satisfy Definition 2, it is worth studying: The effort to use this approach to enable link traversal based query execution in existing SPARQL query engines is comparably small, considering that the majority of engines use an iterator based execution strategy. Furthermore, its limitation may be accepted as a trade-off to avoid inapplicable long query execution times. Various Linked Data based applications employ the approach successfully; e.g., Researchers Map [7], Foaf Letter, AltMed[6], and an approach to consume distributed provenance traces [8].

3.3 The Impact of the Evaluation Order on Query Results

As a consequence of using a fixed evaluation order, the order which is actually being used influences which reachable descriptor objects a query engine discovers and, thus, which solutions it reports. The following example illustrates this effect:

```
?x rdf:type <http://.../X> .
?x ex:p1 ?y .
?y rdfs:label ?z.
?y ex:p2 <http://.../a> .
```

Fig. 1. A sample BGP

Example 1. To execute the BGP in Figure 1 we may select a query plan that uses the order given in the figure. During the execution of this plan the second iterator I_2 requests the first intermediate solution from its predecessor I_1. I_1 ensures that the query-local dataset contains the descriptor object $D_X = lookup($`http://.../X`$)$ [7] and, thereafter, I_1 tries to find matching triples for its triple pattern (i.e. the first pattern in Figure 1). Unfortunately, D_X does not contain such triples (cf. Figure 2). Hence, I_1 cannot

[6] Find AltMed and Foaf Letter as part of `http://www.linkeddata-a-thon.com`

[7] For the sake of simplicity we assume the URIs at the predicate positions resolve to vocabulary definitions that do not contain relevant triples for our example.

Some of the data in $D_x = lookup($`http://.../X`$)$:

<http://.../X> rdfs:subClassOf <http://.../Y> .

Some of the data in $D_b = lookup($`http://.../b`$)$:

<http://.../b> rdfs:label "..." .
<http://.../c> ex:p1 <http://.../b> .

Some of the data in $D_a = lookup($`http://.../a`$)$:

<http://.../b> ex:p2 <http://.../a> .

Some of the data in $D_c = lookup($`http://.../c`$)$:

<http://.../c> rdf:type <http://.../X> .

Fig. 2. Example descriptor objects and some of their data

provide an intermediate solution and, thus, the overall query result is empty. Even if we initialize the query-local dataset with all descriptor objects available from looking up the URIs in the query, i.e. D_X and $D_a = lookup($`http://.../a`$)$, we cannot find a matching triple for the first triple pattern. However, an alternative query plan could use the reverse order, i.e the first iterator is responsible for the last triple pattern in Figure 1. Executing this plan would result in *one* solution for the BGP: $\mu = \{(?\texttt{x}, \texttt{http://.../c}), (?\texttt{y}, \texttt{http://.../b}), (?\texttt{z}, \texttt{"..."})\}$.

As can be seen from the example, the iterator based approach may return different result sets for the same BGP depending on the evaluation order of the triple patterns in the BGP. This effect can be attributed to *missing backlinks* and *serendipitous discovery*, as we discuss in the following.

On the traditional, hypertext Web it is unusual that Web pages are linked bidirectionally. Similarly, an RDF triple of the form (uri_s, uri_p, uri_o) contained in $lookup(uri_s)$ (or $lookup(uri_o)$) does not have to be contained $lookup(uri_o)$ (or $lookup(uri_s)$). We speak of a *missing backlink*. Due to missing backlinks it is possible that one evaluation order allows for the discovery of a matching triple whereas another order misses that triple. For instance, the reason for the different results in Example 1 is a missing backlink in D_X.

Following our look-up assumption each iterator retrieves descriptor objects because these objects may contain matching triples for the triple pattern tp'_i currently evaluated by the iterator. Thereby, all iterators augment the same query-local dataset. Thus, even if retrieved for the evaluation of a specific triple pattern such a descriptor object may also contain a triple t^* that matches another triple pattern tp'_j which will be evaluated later by any of the iterators. Since it is not guaranteed that the descriptor object with t^* is discovered and retrieved during the evaluation of tp'_j, we say that the solution generated based on t^* has been discovered by *serendipity*. If the BGP was ordered differently the descriptor object with t^* might only be discovered after tp'_j has already been evaluated and we could never generate the serendipitously discovered solution.

The effect of missing backlinks and serendipitous discovery on the number of query results is not a characteristic of link traversal based query execution in general; instead, it is specific to the iterator based implementation. In fact, this effect is a direct consequence of the restrictions discussed Section 3.2.

The dependency of result completeness on the order of a BGP implicates that certain orders are more suitable than others. Even if the iterator approach can not be guaranteed to return all solutions that satisfy Definition 2, selecting specific orders could provide for more solutions than other orders. In the next section we discuss the selection of execution orders.

4 Logical Query Planning

In this paper we understand an ordered BGP as a logical query plan. Basically, the creation of such a plan is the selection of a specific order for the triple patterns in a given BGP. Since there exist multiple orders it is possible to create different plans for the same BGP. In this section we discuss how to select one of these plans for the execution of the BGP: We consider the possibility to assess and rank query plans and argue that ranking-based plan selection is unsuitable in our scenario. As a consequence we propose a heuristic for plan selection.

4.1 Assessment of Query Plans

The query plans for a BGP have different characteristics, resulting in different execution performance. We assess them based on two criteria: cost and benefit.

The *cost* of a query plan can be measured in terms of, e.g., query execution time, the amount of network traffic caused, the number of URIs looked up, or the overall size of retrieved descriptor object. We use the query execution time to measure the cost because it provides for a more response time oriented plan assessment and it implicitly includes many of the other measures. For instance, the query execution time is dominated by delays resulting from the look-up of URIs, which may require a significant amount of time due to network latencies. While we propose approaches to reduce the impact of these delays [2], this impact can only be reduced but never be eliminated. Another factor that affects query execution time is the amount of retrieved data: With an increasing number of descriptor objects the time to find matching triples in the query-local dataset may increase, in particular, if each descriptor object is indexed separately [6].

Usually, traditional query optimization uses cost as the only selection criteria for query plans. However, in contrast to traditional query execution, the link traversal based execution of different plans for the same BGP may result in solution sets of different cardinality as we discuss in Section 3.3. Hence, for our iterator approach we should assess query plans not only based on their cost; instead, they must also be assessed by their *benefit*, that is, the number of solutions that an execution of the plan returns.

To rank and select query plans it is necessary to assess each of them without executing it. Since cost (and benefit) cannot be measured without execution, traditional query optimization techniques apply functions that calculate (or estimate) such measures. In our case it would be necessary to take the whole plan into account for such a calculation: The evaluation order of all triple patterns determines what intermediate solutions μ' an iterator I_i consumes from its direct predecessor, what triple patterns $\mu'[tp_i]$ it has to evaluate, what URIs it has to look up and, thus, which reachable descriptor objects it discovers. In this context it is important to note that even the construction of those intermediate solutions which cannot be used for the construction of solutions by subsequent iterators might be beneficial: These solution mappings might be necessary to discover descriptor objects that contribute to completely different solutions as the discussion of serendipitous discovery illustrates (cf. Section 3.3). Notice, due to these dependencies it is impossible to apply the popular dynamic programming approach [9] to generate optimal query plans.

We do not propose actual functions to calculate (or estimate) cost and benefit. Such a calculation requires information about reachable data and the data sources involved in the execution of a plan. In our scenario of link traversal based query execution we do not assume any of such information. We just have a query and an empty local dataset. Hence, before we start executing the query, we do not know anything about the descriptor objects we will discover; we do not even know what descriptor objects will be discovered. Based on this complete lack of information we could only assume a uniform distribution of input values for a cost (or benefit) function. The consequence would be an equal ranking of all possible query plans so that we could at best select a random plan. For this reason we propose to use a heuristic based approach that allows us to make at least an educated guess. However, we note that after starting the query execution it becomes possible to gather information and observe the behavior of the selected plan. This may allow the query system to reassess candidate plans and, thus, to adapt or even replace the running plan. While we do not discuss such a strategy in this paper we will investigate adaptive query planning in the future.

4.2 Heuristic Based Plan Selection

Due to the complete lack of information at plan selection time the application of a cost (and benefit) based ranking of plans is unsuitable in our scenario. For this reason we propose to select query plans based on the following four rules:

- Dependency Respect Rule: Use a dependency respecting query plan.
- Seed TP Rule: Use a plan with a seed triple pattern.
- No Vocab Seed Rule: Avoid a seed triple pattern with vocabulary terms.
- Filtering TP Rule: Use a plan where all filtering triple patterns are as close to the seed triple pattern as possible.

These rules are based on our experience with the data that is currently available as Linked Data, on analyses of the queries executed with our prototypical query engine, and on our experience developing applications that use our query engine. In the remainder of this section we introduce and motivate these rules.

The Dependency Respect Rule proposes to use a *dependency respecting query plan*, that is, an ordered BGP in which at least one of the query variables in each triple pattern occurs in one of the preceding triple patterns. Formally, an ordered BGP $\bar{b} = [tp_1, \dots, tp_n]$ is dependency respecting iff for each $i \in \{2, \dots, n\}$ it holds: $\exists\, v \in vars(tp_i) : \big(\exists\, j < i : v \in vars(tp_j)\big)$. For BGPs which represents a connected graph[8] it is always possible to find a dependency respecting query plan. For the sake of simplicity, we assume all BGPs represent a connected graph.

Dependency respect is a reasonable requirement for query plans in our context because it enables each iterator to always reuse some of the bindings in

[8] A BGP $b = \{tp_1, \dots, tp_n\}$ represents a connected graph iff it holds: $\forall b_1, b_2 \subset b : b_1 \cup b_2 = b \wedge b_1 \cap b_2 = \varnothing \wedge (\exists tp_i \in b_1, tp_j \in b_2 : vars(tp_i) \cap vars(tp_j) \neq \varnothing)$.

intermediate solutions μ' consumed from their predecessor iterator. This strategy avoids what can be understood to be an equivalent to the calculation of cartesian products in RDBMS query executions.

The SEED TP RULE proposes to use a plan in which the first triple pattern is one of the *potential seed triple pattern*s, that are, triple patterns in the BGP which contain at least one HTTP URI. The rationale for using one of the potential seed triple patterns as the first pattern of a plan –which we then call the *seed triple pattern* of that plan– is the following: While query execution begins with an empty query-local dataset, any HTTP URI contained in the query may serve as a starting point to find matching triples. According to our look-up assumption (cf. Section 3.1), matching triples for a triple pattern might be found, in particular, in descriptor objects that were retrieved by looking up the URIs that are part of this pattern. Therefore, it is reasonable to select one of the potential seed triple patterns as the first triple pattern in the query plan.

The NO VOCAB SEED RULE proposes to avoid query plans with a seed triple pattern which contains only URIs that identify vocabulary terms. Such a URI can be identified with high likelihood by a simple syntactical analysis of a triple pattern: Since URIs in the predicate position are always vocabulary terms a preferred seed triple pattern must contain a URI in subject or object position. However, in triple patterns with a predicate of `rdf:type` a URI in the object position always identifies a class, i.e., also a vocabulary term. Hence, these triple patterns should also be avoided as seed triple patterns.

By narrowing down the set of query plans using the NO VOCAB SEED RULE we expect to increase the average benefit of the remaining set of plans. This expectation is based on the following observation: URIs which identify vocabulary terms resolve to RDF data that usually contains vocabulary definitions and very little or no instance data. However, according to our experience the majority of queries asks for instance data and does not contain patterns that have to match vocabulary definitions. Hence, it is reasonable to avoid seed triple patterns that are unlikely to link to instance data as a starting point for query execution. Example 1 illustrates the negative consequences of ignoring the NO VOCAB SEED RULE by selecting the `rdf:type` triple pattern as seed. Notice, for applications that mainly query for vocabulary definitions the rule must be adjusted.

The FILTERING TP RULE proposes to prefer query plans in which filtering triple patterns are placed as close to the seed triple pattern as possible. A *filtering triple pattern* in an ordered BGP contains only query variables that are also contained in at least one preceding triple pattern. Formally, a triple pattern tp_i in an ordered BGP $\bar{b} = [tp_1, \ldots, tp_n]$ is a filtering triple pattern iff it holds: $\forall\, v \in vars(tp_i) : \big(\exists\, j < i : v \in vars(tp_j)\big)$.

The rationale of the FILTERING TP RULE is to reduce cost: During query execution, each intermediate solution μ' consumed by an iterator that is responsible for a filtering triple pattern tp_F, is guaranteed to contain bindings for all variables in tp_F. Therefore, the application of these μ' to tp_F will always result in a triple pattern without variables, i.e. an RDF triple. If this triple is contained in the query-local dataset, the iterator simply passes on the current μ'; otherwise, it discards this intermediate solution. Thus, the evaluation of filtering

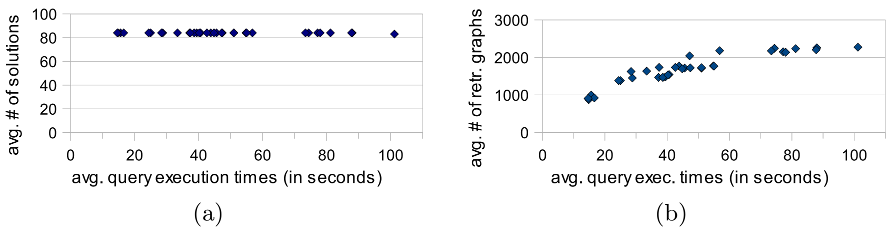


Fig. 3. Measurements for all query plans executed for the "Mylan" query in Figure 4

triple patterns may reduce the number of intermediate solutions but it will never multiply this number. Notice, for other triple patterns we cannot predict such a behavior, neither a reduction nor a multiplication of intermediate solutions.

The Filtering TP Rule is similar to the selection push-down that RDBMSs use to reduce the cost of query plans. However, for link traversal based query execution this rule might not always be beneficial because it reduces the likelihood for serendipitous discovery of matching triples and, thus, solutions (cf. Section 3.3). However, during the evaluation of our heuristic on the current Web of Data we did not experience such a hypothetical reduction of benefit.

5 Experimental Evaluation

In the previous section we argue that cost based plan selection is unsuitable for our iterator implementation of link traversal based query execution. As an alternative we propose a heuristic to select plans. In this section we evaluate the effectiveness of this heuristic.

5.1 Setup

For the evaluation we use three representative BGP queries. None of these queries can be answered using data from a single data provider alone. For each query we generated all dependency respecting query plans that have a seed triple pattern. We executed all these plans sequentially, using a new, initially empty query-local dataset for each plan. For each plan we measured the query execuction time, the number of retrieved descriptor objects, and the number of results. We ran each sequence of plans 6 times where the first run was for warm-up and was not considered for the measurements. These warm-up runs avoid measuring the effect of Web caches and enable servers that contributed data discovered during query execution to adapt their caches. The measurements of the other 5 runs were combined by calculating the arithmetic mean in order to minimize the potential for tampering the experiment by unexpected network traffic.

We executed the query plans using SQUIN[9] which is a prototype of a query engine that implements link traversal based query execution using the presented

[9] http://squin.org

iterator approach. Retrieved descriptor objects are stored separately, each in a main memory index that contains six hashtables to support the typical access patterns[10] during triple pattern matching [6]. The experiment was conducted on an Intel Core 2 Duo T7200 processor with 2 GHz, 4 MB L2 cache, and 2 GB main memory. This machine was connected through the university LAN. Our test system runs a recent 32 bit version of Gentoo Linux with Sun Java 1.6.0.

5.2 Results

For the discussion of our measurements we mainly focus on the query in Figure 4. Nonetheless, we point out notable findings from the measurements taken for the other queries. The BGP in the query in Figure 4 contains 8 triple patterns; one of them qualifies as seed triple pattern according to the No Vocab Seed Rule. For this BGP exist 35 different, dependency respecting query plans with seed triple pattern, each of which has the `diseasome:name` triple pattern as filtering triple pattern.

```
SELECT ?cn ?bd2 WHERE {
  dailymed_orga:Mylan_Pharmaceuticals_Inc .
           dailymed:producesDrug ?bd .
  ?bd dailymed:genericDrug ?gd .
  ?gd drugbank:possibleDiseaseTarget ?dt .
  ?dt diseasome:name "Epilepsy" .
  ?bd dailymed:activeIngredient ?ai .
  ?bd2 dailymed:activeIngredient ?ai .
  ?c dailymed:producesDrug ?bd2 .
  ?c rdfs:label ?cn . }
```

Fig. 4. A SPARQL BGP query (prefix decl. omitted) that asks for companies which use the active ingredient of Mylan Pharmaceuticals' anti-epilepsy drug in their drugs as well

Figure 3(a) illustrates the relationship between the average query execution times (QET) and the average number of solutions measured for the 35 plans. Each point in the chart represents a single plan. As can be seen from the chart, the number of discovered solutions, 84, was the same for all plans. However, the time required to determine these solutions differs significantly. To investigate the reasons that caused these differences we cluster the 35 plans by their QET into 4 groups. Table 1(a) summarizes statistics for these groups: the interval of QET that defines each group (*QET interval*), the number of plans in each group (*# of plans*), the arithmetic mean of the average number of descriptor objects retrieved by each plan in the group (*avg.#DO*), and the arithmetic mean of the position of the filtering triple pattern in each plan of the group (*avg.fTPpos*).

The *avg.fTPpos* values confirm the effectiveness of our heuristic, in particular, the Filtering TP Rule: The less efficient plans in groups G_3 and G_4 contain the filtering triple pattern in the seventh or eighth position; for the more efficient plans in group G_1 it is the fourth or fifth position, hence, closer to the seed triple pattern at the first position. The *avg.#DO* values indicate that during the execution of the less efficient plans more descriptor objects have been retrieved than for the efficient plans. Figure 3(b) illustrates this observation in more detail by representing the plans individually. These measurements

[10] Subject given, predicate given, object given, subj.+pred., subj.+obj., pred.+obj.

Table 1. Statistics about groups of query plans for (a) the "Mylan" query in Figure 4 and (b) the "drug picture" query in Figure 5(a), grouped by the QET of the plans

(a)

	G_1	G_2	G_3	G_4
QET interval	< 20s	[20s,35s]	[35s,60s]	> 60s
# of plans	4	5	18	8
avg.#DO	923.9	1494.0	1695.8	2208.5
avg.fTPpos	4.8	6.0	7.5	7.6

(b)

	G_1	G_2	G_3
QET interval	< 50s	[50s,100s]	> 100s
# of plans	4	3	3
avg.#DO	375.3	496.3	490.9
avg.fTPpos	4.8	6.0	6.0

support the assumptions that motivated the proposal of the Filtering TP Rule: The higher number of discovered descriptor objects indicates that more intermediate solutions have been processed by the less efficient plans.

For the other two queries we observed basically the same behavior. The second query, shown in Figure 5(a), contains 6 triple patterns; one qualifies as seed triple pattern according to the No Vocab Seed Rule. 10 dependency respecting query plans are possible, containing one filtering triple pattern each. The difference between the number of descriptor objects retrieved by the more and the less efficient plans is not as significant for this query as for the other queries (cf. the *avg.#DO* values in Table 1(b) and the corresponding chart in Figure 6). We attribute this to a low selectivity of the filtering triple pattern. The third query, shown in Figure 5(b), contains 7 triple patterns of which two qualify as seed following the No Vocab Seed Rule. There are 56 dependency respecting query plans that can be grouped into two subsets, according to the selected seed triple pattern. Interestingly, all plans in one of these groups did not provide any solutions (cf. Figure 7). An investigation reveals that this problem can be attributed to a missing backlink; this backlink has not be discovered by starting the query execution with the seed triple pattern selected for all plans in the corresponding group. It was impossible to anticipate this problem automatically before executing the query. However, the plans that determined solutions exhibit the expected behavior (cf. Table 2). As the only difference to the other two queries we note that each plan contains two filtering triple patterns.

Table 2. Statistics about the query plans for the "American Badger" query in Figure 5(b), grouped by the QET of the plans. The additional lines in this table list: the arithmetic mean of the average number of solutions determined by each plan in a group (*avg.#Sol*), the arithmetic mean of the triple patterns before the first filtering triple pattern in each plan of a group (*avg.b4fTP1*), the arithmetic mean of the triple patterns after the second filtering triple pattern in each plan of a group (*avg.afTP2*), and the arithmetic mean of the triple patterns between the two filtering triple patterns in each plan of a group (*avg.fTP1Δ2*).

	G_1	G_2	G_3	G_4
QET interval	<20s	[20s,70s]	[70s,130s]	>130s
avg.#Sol	0	0	28.1	27.6
# of plans	30	6	12	8
avg.#DO	13.0	15.6	205.1	309.3
avg.b4fTP1	3.03	3.33	3.67	4.50
avg.afTP2	0.63	0.00	0.67	0.25
avg.fTP1Δ2	1.33	1.67	0.67	0.25

```
SELECT ?gd2 ?p WHERE {
  ?gd db:drugCategory
       drugbank_category:antimalarials .
  ?gd db:brandedDrug ?bd .
  ?c dm:producesDrug ?bd .
  ?c rdfs:label "Pfizer Labs" .
  ?gd owl:sameAs ?gd2 .
  ?gd2 foaf:depiction ?p . }
```

(a)

```
SELECT ?s ?M WHERE {
  geospecies:4qyn7 gs:inFamily ?f .
  ?f skos:narrowerTransitive ?s .
  ?s skos:closeMatch ?m .
  ?m rdfs:subClassOf ?M .
  ?s gs:isExpectedIn ?loc .
  ?loc rdf:type gs:State .
  geospecies:4qyn7 gs:isExpectedIn ?loc .}
```

(b)

Fig. 5. Additional queries (prefix decl. omitted) used for the evaluation: (a) asks for pictures of generic drugs that are categorized as antimalarial and that are drugs, categorized as antimalarial and branded by "Pfizer Labs", (b) asks for species and their genus that are classified in the same family as the American Badger, *Taxidea taxus*, and that are expected in the same states as the American Badger

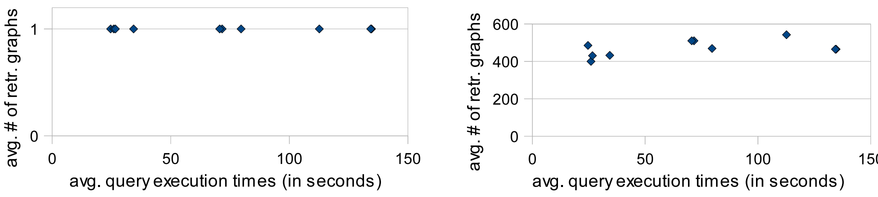


Fig. 6. Measurements for all query plans executed for the query in Figure 5(a)

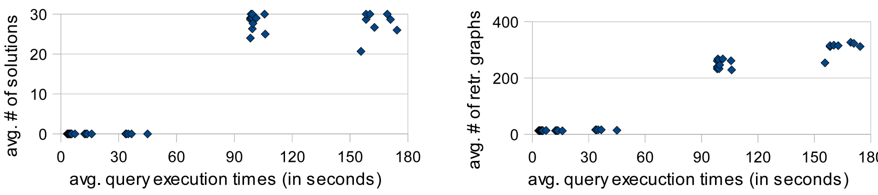


Fig. 7. Measurements for all query plans executed for the query in Figure 5(b)

6 Related Work

In earlier work Mendelzon and Milo introduce an approach to execute SQL-like queries on the traditional, hypertext Web that includes the traversal of links [10]. They formalize the Web as a relational model and propose a two-phase approach to execute queries: First, all "reachable documents are retrieved, and then the query is evaluated on them." The same two-phase approach has been formalized by Bouquet et al. for the Web of Data [11]. While we also use two phases to define solutions in Section 2.2, the idea of link traversal based query execution is to intertwine query evaluation and link traversal instead of simply applying the two-phase approach for the actual execution of queries.

Harth et al. present an alternative approach that also uses URI look-ups to query the Web of data [12]. Instead of traversing links they use a data summary to identify descriptor objects that might be relevant for a query and should be retrieved for the execution. While this approach often performs better than our link traversal approach [3], it requires that all descriptor objects have been discovered, retrieved and summarized before queries can be executed. Furthermore, changes to the data of descriptor objects are not reflected in the summary.

There is only one other implementation approach for link traversal based query execution that we are aware of: In contrast to our synchronized pipeline of iterators, Ladwig and Tran recently proposed an asynchronous implementation that uses symmetric hash joins [3]. While the authors report that their approach returns first results earlier, they measured the same overall query execution times for both approaches. We aim to analyze their approach in the context of the definitions presented here and compare it to our iterator based approach.

7 Conclusion

In this paper we analyze an iterator based implementation approach for the link traversal based query execution paradigm. We study its limitations and discuss how it benefits from a strategy that selects suitable query plans. Such a strategy must work without any statistics, data distribution records or other information about data and data sources because we do not assume any information when we start the execution of a query. Since traditional query planning techniques are unsuitable for this scenario we propose a heuristic for plan selection.

As future work we aim to develop our plan selection rules into a strategy that directly generates the most promising plans only. Furthermore, we investigate how to relax our zero knowledge assumption using information collected during previous query executions. Finally, the integration of adaptive query processing techniques shows great promise to improve our iterator based implementation.

References

1. Hartig, O., Langegger, A.: A database perspective on consuming Linked Data on the Web. Datenbank-Spektrum 10(2) (2010)
2. 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)
3. Ladwig, G., Tran, D.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)
4. Prud'hommeaux, E., Seaborne, A.: SPARQL query language for RDF. W3C Rec. (2008), http://www.w3.org/TR/rdf-sparql-query/
5. Hartig, O.: Iterator based implementations of link traversal based query execution (2010), http://squin.org/doc/IteratorImplementation/
6. Hartig, O.: A main memory index structure to query linked data. In: Proc. of the 4th Int. Linked Data on the Web workshop (LDOW) at WWW (2011)
7. Hartig, O., Mühleisen, H., Freytag, J.C.: Linked Data for building a map of researchers. In: Proc. of 5th Workshop on Scripting and Development for the Semantic Web (SFSW) at ESWC (2009)

8. Hartig, O., Zhao, J.: Publishing and consuming provenance metadata on the Web of Linked Data. In: Proc. of the Int. Provenance and Annotation Workshop (2010)
9. Selinger, P.G., Astrahan, M.M., Chamberlin, D.D., Lorie, R.A., Price, T.G.: Access path selection in a relational database management system. In: Proc. of the Int. Conference on Management of Data (1979)
10. Mendelzon, A.O., Milo, T.: Formal models of Web queries. Information Systems 23(8) (1998)
11. Bouquet, P., Ghidini, C., Serafini, L.: Querying the web of data: A formal approach. In: Gómez-Pérez, A., Yu, Y., Ding, Y. (eds.) ASWC 2009. LNCS, vol. 5926, pp. 291–305. Springer, Heidelberg (2009)
12. Harth, A., Hose, K., Karnstedt, M., Polleres, A., Sattler, K.U., Umbrich, J.: Data summaries for on-demand queries over Linked Data. In: Proc. of the 19th Int. Conference on World Wide Web, WWW (2010)

Integrating Linked Data and Services with Linked Data Services*

Sebastian Speiser and Andreas Harth

Institute AIFB, Karlsruhe Institute of Technology (KIT), Germany
lastname@kit.edu

Abstract. A sizable amount of data on the Web is currently available via Web APIs that expose data in formats such as JSON or XML. Combining data from different APIs and data sources requires glue code which is typically not shared and hence not reused. We propose Linked Data Services (LIDS), a general, formalised approach for integrating data-providing services with Linked Data, a popular mechanism for data publishing which facilitates data integration and allows for decentralised publishing. We present conventions for service access interfaces that conform to Linked Data principles, and an abstract lightweight service description formalism. We develop algorithms that use LIDS descriptions to automatically create links between services and existing data sets. To evaluate our approach, we realise LIDS wrappers and LIDS descriptions for existing services and measure performance and effectiveness of an automatic interlinking algorithm over multiple billions of triples.

1 Introduction

The trend towards publishing data on the Web is gaining momentum, particularly spurred by the Linking Open Data (LOD) project[1] and several government initiatives aimed at publishing public sector data. Data publishers often use Linked Data principles [2]. which leverage established Web standards such as Uniform Resource Identifiers (URIs), the Hypertext Transfer Protocol (HTTP) and the Resource Description Framework (RDF) [9]. Data providers can easily link their data to data from third parties via reuse of URIs. The LOD project proves that the Linked Data approach is, in principle, capable of integrating data from a large number of sources. However, there is still a lot of data residing in silos that could be beneficially linked with other data, but will not be published as a fully materialised knowledge base. Reasons include:

– data is constantly changing, e.g., stock quotes or sensor data can have update intervals below one second;

* This paper is an extension of our previous work [15,16]. We have extended the work with a formal definition of service descriptions, an evaluation of the performance and effectiveness of the proposed methods – including the implementation of several Linked Data Services – and an extensive overview of related work.

[1] http://linkeddata.org/

© Springer-Verlag Berlin Heidelberg 2011

– data is generated depending on possibly infinite different input data, e.g., the distance between two geographical points can be specified with arbitrary precision;
– the data provider does not want arbitrary access to the data, e.g., prices of flight tickets may be only available for specific requests in order to maintain the possibility for price differentiation.

Such data is commonly provided via Web APIs or services, in the following also called data or information services, as they provide a restricted view on a possibly implicit data set. APIs are often based on Representational State Transfer (REST) principles [4], use HTTP as transport protocol and pass parameters as name/value pairs in the URI query string. Currently deployed Web APIs return data as JSON or XML, which requires glue code to combine data from different APIs.

There are useful examples for the integration of information services and Linked Data. Linked Data interfaces for services have been created, e.g., in form of the book mashup [3] which provides RDF about books based on Amazon's API, or twitter2foaf, which encodes a Twitter follower network of a given user based on Twitter's API. However, the interfaces are not formally described and thus the link between services and data has to be established manually or by service-specific algorithms. For example, to establish a link between person instances (e.g., described using the FOAF vocabulary[2]) and their Twitter account, one has to hard-code which property relates people to their Twitter username and the fact that the URI of the person's Twitter representation is created by appending the username to `http://twitter2foaf.appspot.com/id/`.

Vast amounts of idle data can be brought to the Semantic Web via a standardised method for creating Linked Data interfaces to services. The method should incorporate formal service descriptions that enable (semi-)automatic service discovery and integration. We present such an approach for what we call LInked Data Services (LIDS). Specifically, we present the following contributions:

– an access mechanism for LIDS interfaces based on generic Web architecture principles (URIs and HTTP) (Section 3);
– a generic lightweight data service description formalism, instantiated for RDF and SPARQL graph patterns (Section 4);
– an algorithm for linking existing data sets using LIDS (Section 5)

In Section 6 we describe the creation of LIDS for existing services, and present the results of an experiment measuring performance and effectiveness of the approach. The experiment interlinks the 2010 Billion Triple Challenge data set with a geographic LIDS. We relate our approach to existing work in Section 7 and conclude with Section 8.

2 Preliminaries

In the following we shortly present the basics for our work, namely: data services, and RDF.

[2] `http://www.foaf-project.org/`

2.1 Data Services

Our notion of data services is as follows:

> Data services return data dynamically derived (i.e., during service call time) from supplied input parameters. Data services neither alter the state of some entity nor modify data. In other words, data services are free of any side effects. They can be seen as data sources providing information about some entity, when given input in the form of a set of name/value pairs. The notion of data services include Web APIs and REST-based services providing output data in XML or JSON.

Data services are related to Web forms or the "Deep Web" [13], but take and provide data rather than free text or documents. For example, the GeoNames `findNearbyWikipedia` service relates given latitude/longitude parameters to Wikipedia articles describing geographical features that are nearby.

Table 1. Example data-providing services

API	Format	Description
GeoNames	XML, JSON	Functions include besides others: (i) find the nearest GeoNames feature to a given point and (ii) link a geographic point to resources from DBpedia that are nearby URI: `http://www.geonames.org/`
Google GeoCoding API	XML, JSON	Provides latitude and longitude for a given street address. URI: `http://code.google.com/apis/maps/`
Twitter API	XML, JSON, RSS, Atom	Various functions, giving access to Twitter users, follower networks, and tweets. URI: `http://dev.twitter.com/`

Example 1. In Table 1, we list some popular data-providing services. Taking the Google GeoCoding API, to get the geographical coordinates for Karlsruhe, we retrieve the URI `http://maps.googleapis.com/maps/api/geocode/json?address=Karlsruhe&sensor=false`, with the following (abbreviated) result:

```
{ "status": "OK",
  "results": [ {
  ...
  "formatted_address": "Karlsruhe, Germany",
  ...
  "geometry": {
    "location": {
      "lat": 49.0080848,
      "lng": 8.4037563
    },
    ...
  } } ] }
```

Using the retrieved coordinates, we can build the URI for calling the GeoNames service to find Wikipedia articles about things, that are nearby Karlsruhe: `http://ws.geonames.org/findNearbyWikipedia?lat=49.0080848&lng=8.4037563`. The (abbreviated) result is the following:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<geonames>
  <entry>
    <lang>en</lang>
    <title>Federal Constitutional Court of Germany</title>
    ...
    <lat>49.0125</lat>
    <lng>8.4018</lng>
    <wikipediaUrl>...</wikipediaUrl>
    ...
  </entry>
  <entry>
    ...
  </entry>
</geonames>
```

This simple example shows that integrating data from several (in this case only two) services is difficult for the following reasons:

– different serialisation formats are used (e.g., JSON, XML);
– entities are not represented explicitly, and are thus difficult to identify between different services. For example, the geographical point returned by the GeoCoding API does not occur in the output of the GeoNames service. Therefore it is not possible to link the results based on the service outputs alone, but only with service-specific gluing code.

2.2 RDF and Basic Graph Patterns

In contrast to XML or JSON, the Resource Description Framework (RDF) is a graph-based data format which allows for easy integration of data from multiple sources. We now introduce basic RDF notions later reused in the paper; cf. [6].

Let U, B, L, V be disjoint infinite sets of URIs, blank nodes, literals and variables.

Definition 1. *(Triple) A triple $t = (s, p, o)$ is a tuple of length three, $t \in (U \cup B) \times U \times (U \cup B \cup L)$. We often write t as* `s p o`*, where* `s` *is called the subject,* `p` *the predicate and* `o` *the object.*

Definition 2. *(RDF Graph) An RDF graph r is a finite set of triples.*

We often write a set of triples by separating triples by `.` (a dot). To be able to query graphs, we introduce the notion of triple pattern which can include variables.

Definition 3. *(Triple Pattern) A triple pattern* $t \in (U \cup B \cup V) \times (U \cup V) \times (U \cup B \cup L \cup V)$ *abstracts from single triples by allowing variables in every position.*

Definition 4. *(Basic Graph Pattern (BGP) and Conjunctive Query (CQ)) A BGP is a finite set of triple patterns. A conjunctive query* $CQ = (X, T)$ *consists of a head, i.e. a set of variables* $X \subset V$*, and a body, i.e. a BGP* T.

Let M be the set of all function $\mu : U \cup L \cup V \rightarrow U \cup L$, s.t. μ is the identity for constants, i.e. $\forall a : (a \in U \cup L \rightarrow \mu(a) = a)$. As an abbreviation we also apply a function $\mu \in M$ to a triple pattern $t = p(t_1, \ldots, t_n)$ $(\mu(t) = p(\mu(t_1), \ldots, \mu(t_n)))$, and to a BGP T $(\mu(T) = \{\mu(t) \mid t \in T\})$.

Definition 5. *(Variable Binding) A function* $\mu \in M$ *is a variable binding for a conjunctive query* $CQ = (X, T)$ *and a RDF graph* r*, if* $\mu(T) \subseteq r$*. We denote the set of all mappings for a CQ and a graph as* $\mathcal{M}_{CQ}(r) = \{\mu \in M | \mu(T) \subseteq r\}$.

3 Linked Data Services

Linked Data Services provide a Linked Data interface for data services. To make these services adhere to Linked Data principles a number of requirements have to be fulfilled:

- the input for a service invocation with given parameter bindings must be identified by a URI;
- resolving that URI must return a description of the input entity, relating it to the service output data;
- the description must be returned in RDF format.

We call such services *Linked Data Services (LIDS).*

Example 2. Inputs for the LIDS version of the `findNearbyWikipedia` service are entities representing geographical points given by latitude and longitude, which are encoded in the URI of an input entity. Resolving such an input URI returns a description of the corresponding point, which relates it to Wikipedia articles which are nearby.

Defining that the URI of a LIDS call identifies an input entity is an important design decision. Compared to the alternative – directly identifying output entities with service call URIs – identifying input entities has the following advantages:

- the link between input and output data is made explicit;
- one input entity (e.g., a geographical point) can be related to several results (e.g., Wikipedia articles);
- the absence of results can be easily represented by an description without further links;
- the input entity has a constant meaning although data can be dynamic (e.g., the input entity still represents the same point, even though a subsequent service call may relate the input entity to new or updated Wikipedia articles).

More formally we characterise a LIDS by:

– Linked Data Service endpoint: ep, an HTTP URI.
– Local identifier i for the input entity of the service.
– Inputs X_i: names of parameters.

The URI of a service call for a parameter assignment μ (mapping X_i to corresponding values) is constructed in the following way (where addition is understood as string concatenation and subtraction removes the corresponding suffix if it matches):

$$uri(ep, X_i, \mu) = ep + "\texttt{?}" + \sum_{x \in X_i} (x + "\texttt{=}" + \mu(x) + "\texttt{\&}") - "\texttt{\&}"$$

Additionally we introduce an abbreviated URI schema that can be used if there is only one required parameter (i.e. $|X_i| = 1, X_i = \{x\}$):

$$uri(ep, X_i, \mu) = ep + "\texttt{/}" + \mu(x)$$

Please note that the above definition coincides with typical Linked Data URIs. The input entity described by the output of a service call is defined as $inp(ep, X_i, \mu, i) = uri(ep, X_i, \mu) + "\texttt{\#}" + i$.

Example 3. We illustrate the principle using the openlids.org wrapper for GeoNames[3] `findNearbyWikipedia`. The wrapper is a LIDS, defined by:

– endpoint $ep =$ `gw:findNearbyWikipedia`;
– local identifier $i = "point"$;
– inputs $X_i = \{"lat", "lng"\}$.

For a binding $\mu = \{lat \mapsto 49.01, lng \mapsto 8.41\}$ the URI for the service call is `gw:findNearbyWikipedia?lat=49.01&lng=8.41` and returns the following description:

```
@prefix dbpedia: <http://dbpedia.org/resource/> .
gw:findNearbyWikipedia?lat=49.01&lng=8.41#point
    foaf:based_near dbpedia:University_of_Karlsruhe_%28TH%29;
    foaf:based_near dbpedia:Federal_Constitutional_Court_of_Germany;
    foaf:based_near dbpedia:Federal_Court_of_Justice_of_Germany;
    foaf:based_near dbpedia:Wildparkstadion;
    foaf:based_near dbpedia:Karlsruhe.
```

4 Describing Linked Data Services

In this section, we define an abstract model of LIDS descriptions.

[3] `http://km.aifb.kit.edu/services/geowrap/`, abbreviated as `gw`. All other prefixes can be looked up at `http://prefix.cc/`

Definition 6. *(LIDS Description) A LIDS description consists of a tuple* (ep, CQ_i, T_o, i) *where ep denotes the LIDS endpoint,* $CQ_i = (X_i, T_i)$ *a conjunctive query to specify the input to the service,* T_o *a basic graph pattern describing the output data of the service, and i the local identifier for the input entity.*

The meaning of *ep* and X_i were already explained in the previous section. We define X_i to be the head of a conjunctive query, whose body specifies the required relation between the input parameters. T_o specifies the minimum output that is returned by the service for valid input parameters. More formally:

- $\mu \in M$ is a valid input, if $\mu \in M_{CQ_i}(r)$, where r is the implicit RDF graph given by all Linked Data;
- for a valid μ, resolving $uri(ep, X_i, \mu)$ returns a graph $D_o \supseteq \{T' \subseteq D_{impl} \mid \exists \mu \in M : \mu(i) = E_s \wedge \mu(T_o) = T'\}$, where D_{impl} is the implicit, potentially infinite data set representing the information provided by the LIDS.

Example 4. We describe the `findNearbyWikipedia` openlids.org wrapper service as (ep, CQ_i, T_o, i) with:
$ep =$ `gw:findNearbyWikipedia`
$CQ_i = (\{\text{lat,lng}\}, \{$ `?point geo:lat ?lat . ?point geo:long ?lng` $\})$
$T_o = \{$`?point foaf:based_near ?feature`$\}$
$i = point$

4.1 Relation to Source Descriptions in Information Integration Systems

Note that the LIDS descriptions can be transformed to source descriptions with limited access patterns, in a Local-as-View (LaV) data integration approach [5]. With LaV, the data accessible through a service is described as a view in terms of a global schema. The variables of a view's head predicate that have to be bound in order to retrieve tuples from the view are prefixed with a \$. For a LIDS description (ep, CQ_i, T_o, i), we can construct the LaV description:

$$ep(\$I_1, \ldots, \$I_k, O_1 \ldots, O_m) \texttt{ :- } p_1^i(\ldots), \ldots, p_n^i(\ldots), p_1^o(\ldots), \ldots, p_l^o(\ldots).$$

Where $CQ_i = (X_i, T_i)$, $X_i = \{I_1, \ldots, I_k\}$, $T_i = \{(s_1^i, p_1^i, o_1^i), \ldots, (s_n^i, p_n^i, o_n^i)\}$, $T_o = \{(s_1^o, p_1^o, o_1^o), \ldots, (s_l^o, p_l^o, o_l^o)\}$, and $vars(T_o) \setminus vars(T_i) = \{O_1, \ldots, O_m\}$.

We propose for LIDS descriptions the separation of input and output conditions for three reasons: (i) the output of a LIDS corresponds to an RDF graph as described by the output pattern, not to tuples as it is common in LaV approaches, (ii) it is easier to understand for users, and (iii) it is better suited for the interlinking algorithm as shown in Section 5.

4.2 Describing LIDS Using RDF and SPARQL Graph Patterns

In the following we present how LIDS descriptions can be represented in RDF, thus enabling that LIDS descriptions can be published as Linked Data. The basic format is as follows (unqualified strings consisting only of capital letters are placeholders and explained below):

```
@prefix lids: <http://openlids.org/vocab#>

LIDS a lids:LIDS;
     lids:lids_description [
         lids:endpoint ENDPOINT ;
         lids:service_entity ENTITY ;
         lids:input_bgp INPUT ;
         lids:output_bgp OUTPUT ;
         lids:required_vars VARS
      ] .
```

The RDF description is related to our abstract description formalism in the following way:

– `LIDS` is a resource representing the described Linked Data service;
– `ENDPOINT` is a URI corresponding to ep;
– `ENTITY` is the name of the entity i;
– `INPUT` and `OUTPUT` are basic graph patterns encoded as a string using SPARQL syntax. `INPUT` is mapped to T_i and `OUTPUT` is mapped to T_o.
– `VARS` is a string of required variables separated by blanks, which is mapped to X_i.

From this mapping, we can construct an abstract LIDS description $(ep, (X_i, T_i), T_o, i)$ for the service identified by `LIDS`.

Example 5. In the following we show the RDF representation of the formal LIDS description from Example 4:

```
:GeowrapNearbyWikipedia a lids:LIDS;
  lids:lids_description [
     lids:endpoint
       <http://km.aifb.kit.edu/services/geowrap/findNearbyWikipedia>;
     lids:service_entity "point" ;
     lids:input_bgp "?point a Point . ?point geo:lat ?lat .
                                      ?point geo:long ?long" ;
     lids:output_bgp "?point foaf:based_near ?feature" ;
     lids:required_vars "lat long"
  ] .
```

In future, we expect a standardised RDF representation of SPARQL, which does not rely on string encoding of basic graph patterns. One such candidate is the SPIN SPARQL Syntax[4], which is part of the SPARQL Inferencing Notation (SPIN)[5]. We are planning to reuse such a standardised RDF representation of basic graph patterns and variables in future versions of the LIDS description model.

[4] http://spinrdf.org/sp.html
[5] http://spinrdf.org/

5 Algorithm for Interlinking Data with LIDS

In the following, we describe how existing data sets can be automatically enriched with links to LIDS, which can happen in different settings. Consider for example:

– processing of a static data set, inserting links to LIDS and storing the new data;
– an endpoint that serves data (e.g., a Linked Data server), and dynamically adds links to LIDS;
– a data browser that locally augments retrieved data with data retrieved from LIDS.

We present an algorithm that, based on a fixed local dataset, determines and invokes the appropriate LIDS and adds the output to the local dataset.

Given an RDF graph r and a LIDS description $l = (ep, CQ_i, T_o)$ the following formula defines a set of entities in r and equivalent entities that are inputs for the LIDS (i is determined from T_i and T_o and $+$ is again string concatenation):

$$equivs_{r,l} = \Big\{ \big(\mu(i), uri(ep, X_i, \mu) + \text{"\#"} + i\big) \mid \mu \in \mathcal{M}_{CQ_i}(r)\} \Big\}.$$

The obtained equivalences can be either used to immediately resolve the LIDS URIs and add the data to r, or to make the equivalences explicit in r, for example, by adding the following triples to r:

$$\big\{x_1 \ \texttt{owl:sameAs} \ x_2 \mid (x_1, x_2) \in equivs_{r,l}\big\}.$$

Based on the services shown in Figure 1 together with descriptions, we illustrate the algorithm using the following example: consider as starting point an entity URI (e.g., an entity `#aifb`), which, when visited, returns an RDF graph with latitude and longitude properties:

```
#aifb
    rdfs:label "AIFB - Building 11.40";
    geo:lat "49.01";
    geo:long "8.41".
```

In the first step, the data is matched against the available LIDS descriptions (for brevity we assume a static set of LIDS descriptions) and a set of bindings are derived. Further processing uses the GeoNames LIDS which accepts latitude/longitude as input. After constructing a URI which represents the service entity, an equivalence (`owl:sameAs`) link is created between the original entity `#aifb` and the service entity:

```
#aifb owl:sameAs
       gw:findWikipediaNearby?lat=49.01&long=8.41#point.
```

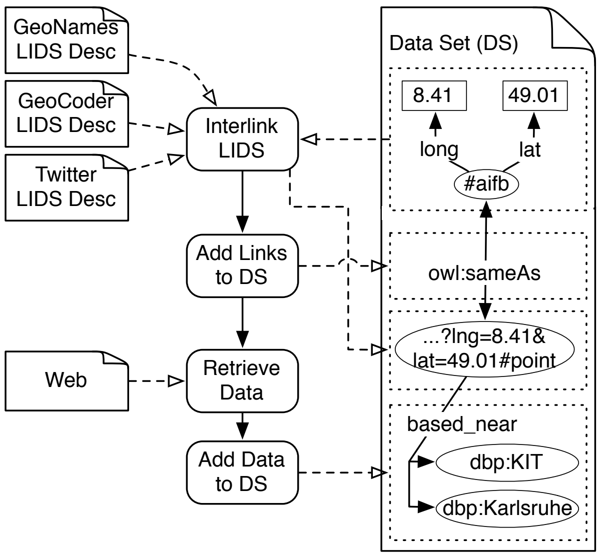


Fig. 1. Interlinking example for GeoNames LIDS

Next, the data from the service entity URI can be retrieved, to obtain the following data:

```
@prefix dbpedia: <http://dbpedia.org/resource/> .
gw:findWikipediaNearby?lat=49.01&long=8.41#point
       foaf:based_near foaf:based_near dbpedia:Wildparkstadion;
       foaf:based_near dbpedia:Karlsruhe.
...
```

Please observe that by equating the URI from the input data with the LIDS entity URI, we essentially add the returned `foaf:based_near` statements to `#aifb`. Should the database underlying the service change, a lookup on the LIDS entity URI returns the updated data which can then be integrated. As such, entity URIs can be linked in the same manner as plain Linked Data URIs.

6 Evaluation of Performance and Effectiveness

We first present several LIDS services which we have made available, and then cover the evaluation of performance and effectiveness of the presented algorithm. Source code and test data for the implementation of the interlinking algorithm, as well as other general code for handling LIDS and their descriptions can be found online[6]. All experiments were conducted on a 2.4 GHz Intel Core2Duo laptop with 4 GB of main memory.

[6] `http://code.google.com/p/openlids/`

6.1 Implemented LIDS Services

In this section, we show how we applied the LIDS approach to construct publicly available Linked Data interfaces for selected existing services.

The following services are hosted on Google's App Engine cloud environment. The services are also linked on `http://openlids.org/` together with their formal LIDS descriptions and further information, such as URIs of example entities.

- GeoNames Wrapper[7] provides three functions:
  - finding the nearest GeoNames feature to a given point,
  - finding the nearest GeoNames populated place to a given point,
  - linking a geographic point to resources from DBpedia that are nearby.
- GeoCoding Wrapper, returning the geographic coordinates of a street address.
- Twitter Wrapper[8] links Twitter account holders to the messages they post.

The effort to produce a LIDS wrapper is typically low. The interface code that handles the service URIs and extracts parameters can be realised by standardised code or even generated automatically from a LIDS description. The main effort lies in accessing the service and generating a mapping from the service's native output to a Linked Data representation. For some services it is sufficient to write XSLTs that transform XML to RDF, or simple pieces of procedural code that transform JSON to RDF. Effort is higher for services that map Web page sources, as this often requires session and cookie handling and parsing of faulty HTML code. However, the underlying data conversion has to be carried out whether or not LIDS are used. Following the LIDS principles is only a minor overhead in implementation; adding a LIDS description requires a SPARQL query to describe the service.

6.2 Interlinking Existing Data Sets with LIDS

We implemented a streaming version of the interlinking algorithm shown in Section 5 based on NxParser[9]. For evaluation of the algorithm's performance and effectiveness we interlinked the Billion Triple Challenge (BTC) 2010 data set[10] with the `findNearby` geowrapper. In total the data set consisted of 3,162,149,151 triples and was annotated in 40,746 seconds ($<$ 12 hours) plus about 12 hours for uncompressing the data set, result cleaning, and statistics gathering. In the cleaning phase we filtered out links to the geowrapper that were redundant, i.e., entities that were already linked to GeoNames, including the GeoNames data set itself. The original BTC data contained 74 different domains that referenced GeoNames URIs. Our interlinking process added 891 new domains that are now linked to GeoNames via the geowrap service. In total 2,448,160 new links were

[7] `http://km.aifb.kit.edu/services/geowrap/`
[8] `http://km.aifb.kit.edu/services/twitterwrap/`
[9] `http://sw.deri.org/2006/08/nxparser/`
[10] `http://km.aifb.kit.edu/projects/btc-2010/`

added[11]. Many links referred to the same locations, all in all there were links to ca. 160,000 different geowrap service calls. These results show that even with a very large data set, interlinking based on LIDS descriptions is feasible on commodity hardware. Furthermore, the experiment showed that there is much idle potential for links between data sets, which can be uncovered with our approach.

7 Related Work

Our work provides an approach to open up data silos for the Web of Data. Previous efforts in this direction are confined to specialised wrappers, for example the book mashup [3]. Other state-of-the-art data integration systems [18] use wrappers to generate RDF and then publish that RDF online rather than providing access to the services that generate RDF directly. In contrast to these ad-hoc interfaces, we provide a uniform way to construct such interfaces, and thus our work is applicable not only to specific examples but generally to all kinds of data silos. Furthermore, we present a method for formal service description that enables the automatic interface generation and service integration into existing data sets.

SILK [19] can be used to discover links between Linked Data from different sources. Using a declarative language, a developer specifies conditions that data from different sources has to fulfill to be merged, optionally using heuristics in case merging rules can lead to ambiguous results. In contrast, we use Linked Data principles for exposing content of data-providing services, and specify the construction of URIs which can be related to already existing data.

There exists extensive literature about semantic descriptions of Web services. We distinguish between two kinds of works: (i) general semantic Web service (SWS) frameworks, and (ii) stateless service descriptions.

General SWS approaches include OWL-S [11] and WSMO [14] and aim at providing extensive expressivity in order to formalise every kind of Web service, including complex business services with state changes and non-trivial choreographies. The expressivity comes at a price: SWS require complex modeling even for simple data services using formalisms that are not familiar to all Semantic Web developers. In contrast, our approach focuses on simple data services and their lightweight integration with Linked Data.

Most closely related to our service description formalism are works on semantic descriptions of stateless services (e.g., [8,7,20]). Similar to our approach these solutions define service functionality in terms of input and output conditions. Most of them, except [8], employ proprietary description formalisms. In contrast, our approach relies on standard SPARQL. Moreover, our work provides the following key advantages: (i) a methodology to provide a Linked Data interface to services, (ii) semi-structured input and output definitions, compared to the static definition of required inputs and outputs in previous approaches.

[11] Linking data is available online: `http://people.aifb.kit.edu/ssp/geolink.tgz`

Norton and Krummenacher propose an alternative approach to integrate Linked Data and services, so-called Linked Open Services (LOS) [12]. LOS descriptions also use basic graph patterns for defining service inputs and outputs. One difference to our work is that LOS consume RDF instead of name-value pairs. With the LIDS approach, service calls are directly linkable from within Linked Data, as service inputs are encoded in the query string of a URI.

Other related work to integrating data comes from the database community, specifically information integration. Mediator systems (e.g., Information Manifold [10]) are able to answer queries over heterogeneous data sources, including services on the Web. Information-providing data services were explicitly treated, e.g., in [17,1]. For an extensive overview of query answering in information integration systems, we refer the reader to [5]. All these works have in common that they answer queries using services, but do not provide methods to expose services with a standardised interface and link-able interfaces. Thus information integration is only done at the time of query answering, which is in contrast to our proposed approach that allows data sets to be directly interlinked, independent of a query processor.

8 Conclusions

A large portion of data on the Web is attainable through a large number of data services with a variety of interfaces that require procedural code for the integration of different data sources. We presented a general method for exposing data services as Linked Data, which enables the integration of different data sources without specialised code. Our method includes an interface convention that allows service inputs to be given as URIs and thus linked from other Linked Data sources. By exposing URIs for service inputs in addition to service outputs, the model neatly integrates with existing data, can handle multiple outputs for one input and makes the relation between input and output data explicit.

Furthermore, we proposed a lightweight description formalism and showed how it can be used for automatically interlinking Linked Data Services with appropriate data sets. We showed how the descriptions can be instantiated in SPARQL. We applied our method to create LIDS for existing real-world service, thus contributing new data to the Web. The approach was evaluated for performance and effectiveness in an experiment in which we interlinked the Billion Triple Challenge (BTC) 2010 data set with the GeoNames LIDS wrapper. We showed that the algorithm scales even to this very large data set and produces large numbers (around 2.5 million) of new links between entities. A possible avenue for future work would be to integrate fuzzy matching algorithms, similar to [19], in case the input to a web service is ambiguous, e.g., for services which take keywords as input.

We further plan future work in three main areas:

– improve tool support, so that Semantic Web developers can easily adopt the LIDS method for their applications and services;

– develop approaches for integrating LIDS into SPARQL query processing;
– integrate provenance information and usage policies in the service descriptions, in order to ensure legal compliance and traceability of integrated data sets.

Acknowledgements

The authors acknowledge the support of the European Community's Seventh Framework Programme FP7/2007-2013 (PlanetData, Grant 257641) and of the Deutsche Forschungsgemeinschaft (Information Management and Engineering Graduate School, GRK 895).

References

1. Barhamgi, M., Champin, P.-A., Benslimane, D.: A Framework for Web Services-Based Query Rewriting and Resolution in Loosely Coupled Information Systems (2007)
2. Berners-Lee, T.: Linked Data. Design Issues (2009), `http://www.w3.org/DesignIssues/LinkedData`
3. Bizer, C., Cyganiak, R., Gauss, T.: The RDF Book Mashup: From Web APIs to a Web of Data. In: Workshop on Scripting for the Semantic Web (2007)
4. Fielding, R.T., Taylor, R.N.: Principled design of the modern web architecture. ACM Trans. Internet Technol. 2, 115–150 (2002)
5. Halevy, A.Y.: Answering queries using views: A survey. The VLDB Journal 10, 270–294 (2001)
6. Hayes, P.: RDF Semantics. W3C Recommendation (February 2004), `http://www.w3.org/TR/rdf-mt/`
7. Hull, D., Zolin, E., Bovykin, A., Horrocks, I., Sattler, U., Stevens, R.: Deciding Semantic Matching of Stateless Services. In: AAAI Conference on Artificial Intelligence, AAAI (2006)
8. Iqbal, K., Sbodio, M.L., Peristeras, V., Giuliani, G.: Semantic Service Discovery using SAWSDL and SPARQL. In: International Conference on Semantics, Knowledge and Grid, SKG (2008)
9. Klyne, G., Carroll, J.J.: Resource Description Framework (RDF): Concepts and Abstract Syntax. W3C Rec. (February 2004), `http://www.w3.org/TR/rdf-concepts/`
10. Levy, A.Y., Rajaraman, A., Ordille, J.J.: Querying Heterogeneous Information Sources Using Source Descriptions. In: International Conference on Very Large Data Bases, VLDB (1996)
11. Martin, D., Burstein, M., Hobbs, J., Lassila, O., McDermott, D., McIlraith, S., Narayanan, S., Paolucci, M., Parsia, B., Payne, T., Sirin, E., Srinivasan, N., Sycara, K.: OWL-S: Semantic Markup for Web Services (2004), `http://www.w3.org/Submission/OWL-S/`
12. Norton, B., Krummenacher, R.: Consuming dynamic linked data. In: First International Workshop on Consuming Linked Data, COLD 2010 (2010)
13. Raghavan, S., Garcia-Molina, H.: Crawling the hidden web. In: International Conference on Very Large Data Bases (VLDB), pp. 129–138 (2001)

14. Roman, D., Keller, U., Lausen, H., de Bruijn, J., Lara, R., Stollberg, M., Polleres, A., Feier, C., Bussler, C., Fensel, D.: Web service modeling ontology. Applied Ontology 1(1), 77–106 (2005)
15. Speiser, S., Harth, A.: Taking the LIDS off Data Silos. In: Triplification Challenge at I-SEMANTICS (2010)
16. Speiser, S., Harth, A.: Towards Linked Data Services. In: The Semantic Web - Posters and Demonstrations, ISWC (2010)
17. Thakkar, S., Ambite, J.L., Knoblock, C.A.: A Data Integration Approach to Automatically Composing and Optimizing Web Services. In: Workshop on Planning and Scheduling for Web and Grid Services (2004)
18. Troncy, R., Fialho, A., Hardman, L., Saathoff, C.: Experiencing events through user-generated media. In: First International Workshop on Consuming Linked Data, COLD 2010 (2010)
19. 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)
20. Zhao, W.-F., Chen, J.-L.: Toward Automatic Discovery and Invocation of Information-Providing Web Services. In: Mizoguchi, R., Shi, Z.-Z., Giunchiglia, F. (eds.) ASWC 2006. LNCS, vol. 4185, pp. 474–480. Springer, Heidelberg (2006)

OntoWiki Mobile – Knowledge Management in Your Pocket

Timofey Ermilov, Norman Heino, Sebastian Tramp, and Sören Auer

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

Abstract. As comparatively powerful mobile computing devices are becoming more common, mobile web applications have started gaining in popularity. In this paper we present an approach for a mobile semantic collaboration platform based on the OntoWiki framework. It allows users to collect instance data, refine the structure of knowledge bases and browse data using hierarchical or faceted navigation on-the-go even without a present data connection. A crucial part of OntoWiki Mobile is the advanced replication and conflict resolution for RDF content. The approach for conflict resolution is based on a combination of distributed revision control strategies and the EvoPat method for data evolution and ontology refactoring. OntoWiki mobile is available as an HTML5 Web application and can be used in scenarios where semantically rich information has to be collected in field-conditions such as during bio-diversity expeditions to remote areas.

1 Introduction

As comparatively powerful mobile computing devices are becoming more common, mobile web applications have started gaining in popularity. Mobile web applications such as *Google Mail* or *Calendar* are already in use everyday by millions of people. Some of these applications already use the Semantic Web technologies and information in the form of RDF (e. g. TripIt). An important feature of these applications is their ability to provide *offline functionality* with local updates for later synchronization with a web server. The key problem here is the reconciliation, i. e. the problem of potentially *conflicting updates* from *disconnected clients*.

Another problem current mobile application developers face is the plethora of mobile application development platforms as well as the incompatibilities between them. *Android* (Google), *iOS* (Apple), *Blackberry OS* (RIM), *WebOS* (HP/Palm), *Symbian* (Nokia) are popular and currently widely deployed platforms, with many more proprietary ones being available as well. As a consequence of this fragmentation, realizing a special purpose application, which works with many or all of these platforms is extremely time consuming and inefficient due to the large amount of duplicate work required.

© Springer-Verlag Berlin Heidelberg 2011

The W3C addressed this problem, by enriching HTML in its 5th revision with access interfaces to local storage (beyond simple cookies) as well as a number of devices and sensors commonly found on mobile devices (e. g. GPS, camera, compass etc.). We argue, that in combination with semantic technologies these features can be used to realize a *general purpose*, mobile collaboration platform, which can support the long tail of mobile special interest applications, for which the development of individual tools would not be (economically) feasible.

In this paper we present the *OntoWiki Mobile* approach realizing a mobile semantic collaboration platform based on the OntoWiki framework [2]. It comprises specifically adopted user interfaces for browsing, faceted navigation as well as authoring of knowledge bases. It allows users to collect instance data and refine the structured knowledge bases on-the-go. OntoWiki Mobile is implemented as an *HTML5 web application*, thus being completely mobile device platform independent. In order to allow offline use in cases with restricted network coverage (or in order to avoid roaming charges) it uses the novel HTML5 local storage feature for replicating parts of the knowledge base on the mobile device. Hence, a crucial part of OntoWiki Mobile is the advanced conflict resolution for RDF stores. The approach is based on a combination of the EvoPat [8] method for data evolution and ontology refactoring along with a versioning system inspired by distributed version control systems like Git.

There are already a number of mobile semantic applications ranging from semantic backend services [11] for mobile devices to applications covering very specific use cases (e. g. *DBpedia Mobile* [1] or *mSpace Mobile* [14]). OntoWiki Mobile, however, is a generic, application domain agnostic tool, which can be utilized in a wide range of very different usage scenarios ranging from instance acquisition to browsing of semantic data on the go. Typical OntoWiki Mobile usage scenarios are settings where users need to author and access semantically structured information on the go or in settings where users are away from regular power supply and restricted to light-weight equipment (e. g. scientific expeditions).

The paper is structured as follows: We outline the general architecture of OntoWiki Mobile in Section 2. We describe our replication and reconciliation strategy in Section 3. The OntoWiki Mobile user interface and the implementation of browsing and authoring in restricted mobile environments is presented in Section 4. A description of a use case for OntoWiki mobile in the domain of field expeditions in bio-diversity research is presented in Section 5. We give an overview on related work in Section 6 and conclude with an outlook on future work in Section 7.

2 Architecture

OntoWiki was developed to address the need for a Web application for rapid and simple knowledge acquisition in a collaborative way. OntoWiki can be used for presenting, authoring and managing knowledge bases adhering to the RDF data model. In order to render its functionality, OntoWiki relies on several APIs

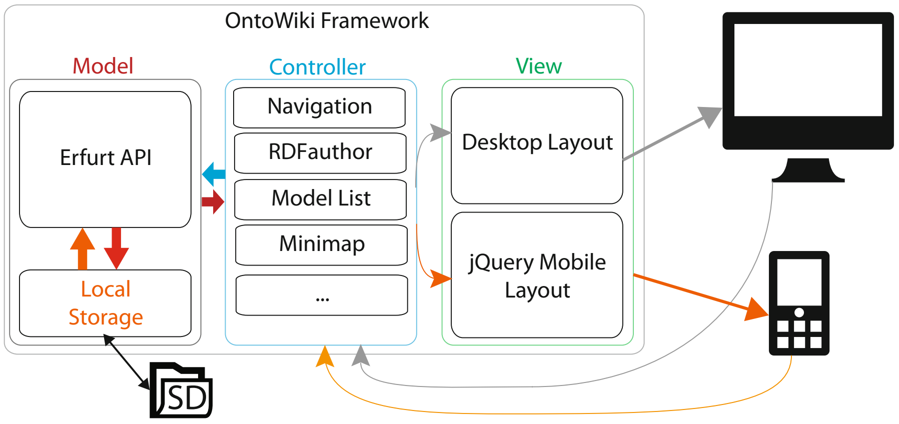


Fig. 1. OntoWiki Mobile architecture

that are also available to third-party developers. Usage of these programming interfaces enables the users to extend, customize and tailor OntoWiki in several ways. OntoWiki's architecture consists of three separate layers: persistence layer, application layer and user interface layer. Those layers represent the standard MVC[1] architecture. The persistence layer consists of the Erfurt API which provides an interface to different RDF stores (e. g. Virtuoso, MySQL). Content in OntoWiki is rendered through the templates (user interface layer). The controller action serving the request renders its output in a template. OntoWiki as a Web application is based on the Zend Framework[2] which lays out the basic architecture and is primarily responsible for request handling. Such architecture allows to easily extend the functionality of OntoWiki and change the layout based on context parameters of the user request. OntoWiki Mobile is based on the OntoWiki Framework. It utilizes all of the described OntoWiki Framework architecture and tailors it to better fit mobile usage scenarios by e. g. replacing with mobile-specific layout (see Figure 1).

The mobile user interface was built using HTML5 and the jQuery Mobile[3] framework which includes the core jQuery library in an improved version to ensure compatibility across all of the major mobile platforms. Built on a jQuery and jQuery UI foundation, it allowed us to create a unified user interface regardless of the actual platform the user's device runs on. The resulting source code presents a thin JavaScript layer, built with Progressive Enhancement principles so as to allow for a minimal footprint.

To access the device's hardware (e. g. camera, GPS sensor) OntoWiki Mobile uses the extended HTML5 API[4]. Geolocation API is used from JavaScript to

[1] Model-View-Controller.

[2] http://framework.zend.com/

[3] http://jquerymobile.com/

[4] http://w3.org/TR/html5/offline.html#offline

get user's current latitude and longitude. Local storage is a part of the HTML5 application caches and is a persistent data storage of key-value pair data in Web clients. It is used to store replicated parts of knowledge bases at the client-side. OntoWiki Mobile stores RDF data as JSON-encoded strings for offline usage and to increase page loading speed while online (e. g. in cases where the resource has not been changed and does not require reloading). Attached photograph are stored to local cache using HTML5 Canvas base64 encoding. Usage of local storage allows to export and import user-gathered data, for example to do back-ups (snapshots) of data to external SD card or to share data with other mobile devices via bluetooth.

Resource editing in OntoWiki is done using RDFauthor [12]. The system makes use of RDFa-annotations in web views in order to make RDF model data available on the client. Embedded statements are used to reconstruct the graph containing statements about the resource being edited. A set of editing widgets, tailored to specific editing tasks and equipped with end-user supporting functionalities (e. g. resource autocompletion from OntoWiki and Sindice) are selected based on the statements contained in the graph. In OntoWiki Mobile, RDFauthor has been adapted to better cope with mobile environments by adapting the user interface and introducing lazy script loading.

Data replication and conflict resolution is the most complex part of the OntoWiki Mobile. The process consists of three steps, handled by separate components (explained in more detail in Section 3):

- a client-side replication component utilizing HTML5 local storage,
- a server-side replication component and
- a server-side conflict resolver.

The conflict resolver uses additional mechanisms to simplify merging concurrent edits of the same resource. The first one is policy-based semi-automatic merging tool that utilizes the EvoPat engine – an OntoWiki extension for dealing with evolution of knowledge bases using patterns [8]. Evolution patterns in EvoPat consists of variables, a SPARQL query template and a SPARQL/Update query with functional extensions. Results of the SPARQL query are bound to variables which in turn are used in SPARQL/Update queries to perform knowledge base transformations. OntoWiki Mobile uses specifically created patterns that can be applied by the user. The second mechanism provides a user interface for manual conflict resolution. It allows the user to select which statements from different version to include in a merged version of a resource.

3 Replication

One critical requirement for OntoWiki Mobile was the ability to work without an Internet connection. In cases where several users edit the same resources without synchronization in between, replication issues may occur. At least one of the users is likely to be working with an outdated version of a resource.

When the user attempts to synchronize data with the main OntoWiki server, several steps are taken to minimize the need for human intervention. However, fully automatic conflict resolving is not possible in all cases.

3.1 Concepts

The unit of editing and display in OntoWiki is a *resource*. Since OntoWiki Mobile needs to identify the same resource at different points in time, we define a resource r_t as a set of triples contained in some graph that share the same subject s at a certain point in time, i. e. a description of r at timestamp t. When an editing operation is carried out, all the changed triples are saved/deleted at once for a given resource. Thus, a *diff* d_{t_1,t_2}, $t_1 < t_2$ is the change applied to a resource description from timestamp t_1 to timestamp t_2. It is defined as a quadruple

$$d_{t_1,t_2} := (t_1, t_2, \mathsf{Add}, \mathsf{Del}) = (t_1, t_2, r_{t_1} \setminus r_{t_2}, r_{t_2} \setminus r_{t_1}).$$

That is, it contains a set of added and a set of removed statements that led from r_{t_1} to r_{t_2}. Two diffs $d_{s_1,t_1} = (s_1, t_1, \mathsf{Add}_1, \mathsf{Del}_1)$ and $d_{s_2,t_2} = (s_2, t_2, \mathsf{Add}_2, \mathsf{Del}_2)$ are said to be *in conflict* if both of the following conditions are met:

$$2 \cdot |\mathsf{Add}_1 \cup \mathsf{Add}_2| > |\mathsf{Add}_1| + |\mathsf{Add}_2| \quad (1)$$

$$\mathsf{Del}_1, \mathsf{Del}_2 \neq \emptyset \Rightarrow \mathsf{Del}_1 \cap \mathsf{Del}_2 \neq \emptyset \quad (2)$$

In other words, both diffs remove at least one identical and add at least one different triple. The empty diff $d_{s,t} = (s, t, \emptyset, \emptyset)$ does not conflict with any other diff. This definition gives necessary, but not sufficient, conditions for conflicting changesets, i. e. there are non-conflicting changesets that meet both conditions.

Let $d_{s_1,t_1} = (s_1, t_1, \mathsf{Add}_1, \mathsf{Del}_1)$ and $d_{s_2,t_2} = (s_2, t_2, \mathsf{Add}_2, \mathsf{Del}_2)$ be two diffs at timestamps t_1 and t_2 with $s_i < t_i$, $t_1 < s_2$. The *concatenation* operation $\circ : \mathsf{D} \times \mathsf{D} \longrightarrow \mathsf{D}$ (D denoting the set of all diffs) yields a new diff with the combined additions and deletions from d_{s_1,t_1} and d_{s_2,t_2}:

$$d_{s_1,t_1} \circ d_{s_2,t_2} = (s_1, t_2, \mathsf{Add}_1 \cup \mathsf{Add}_2, \mathsf{Del}_1 \cup \mathsf{Del}_2).$$

Consecutive diffs to the same resource r_t are combined to a *changeset*s, which are exchanged between mobile devices (OntoWiki Mobile) and OntoWiki on synchonization.

3.2 Synchronization

We are now in the position to specify what happens when users synchronize data with OntoWiki. Given a re-established data connection and the user's consent, OntoWiki Mobile sends all changesets back to the server's synchronization component. Let c be a changeset on resource r with diffs $(d_{s_1,t_1}, d_{s_2,t_2}, \ldots, d_{s_k,t_k})$. OntoWiki Mobile concatenates the diffs contained in c into a single diff d_{s_1,t_k}. A server diff is then calculated as $d_{s,t}$ where s and t are the largest timestamps for changes on r smaller than s_1, t_k respectively. Using conditions (1) and (2) conflicting diffs are determined. In case of a conflict, all diffs $d_{s,t}$ in c are applied sequentially (w. r. t. t) until the conflict occurs. At this point, two branches of r are created having as last common version $r_{t_{k-1}}$ where t_k is the conflicting patch.

For merging branches, two different ways exist: manually (using the OntoWiki's merging UI) or semi-automatic (using EvoPat). EvoPat allows the application of policy-based merging patterns on conflicting branches. Patterns for the following merging policies are provided with OntoWiki Mobile:

– *User privilege-based* – changesets from users with higher priority have prevalence or
– *time privilege-based*, which can also be called "first-come, first-serve" – the latest changes are considered least prioritized.

Additional policies (in the form of EvoPat evolution patterns) can be created by the user, if needed.

There are some situations that cannot be completely resolved without user intervention or creation of additional rules for EvoPat. For example, in cases where two users create a resource describing the same real world object by using different identifiers.

3.3 Example

Consider two users *Alice* and *Bob* who both work on the same resource r_{t_0} which has two statements, s_1 and s_2, as of timestamp t_0. The described scenario is depicted in Figure 2.

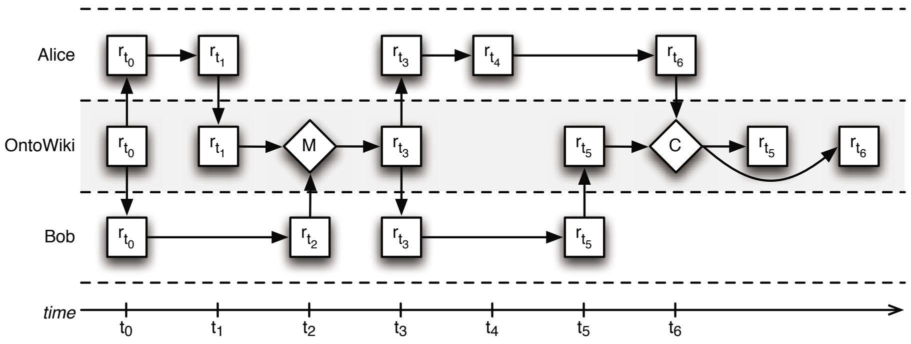


Fig. 2. Data replication example with merge (M) and conflict detection (C)

At t_1, Alice changes statement s_1 to s_3; this is actually reflected as deleting s_1 and adding s_3. In the same way, Bob removes s_2 and adds s_4 at t_2. When both synchronize with OntoWiki, their respective changesets contain only one patch each

$$c_{t_0,t_1,\text{Alice}} = (t_0, t_1, \{s_3\}, \{s_1\}) \quad \text{and} \quad c_{t_0,t_2,\text{Bob}} = (t_0, t_2, \{s_4\}, \{s_2\}).$$

Since $\{s_1\} \cap \{s_2\} = \emptyset$, condition (2) does not hold and we have non-conflicting changesets that can be merged into r_{t_3} at t_3. Alice then adds another statement s_5 at t_4 and later discovers that she entered duplicate information and decides to remove s_3 at t_5. Meanwhile, Bob also notices the error on s_3, removes it and adds s_6 at t_4. This time, when both synchronize their data with OntoWiki, we have patches

$$c_{t_3,t_6,\text{Alice}} = (t_3, t_6, \{s_5\}, \{s_3\}) \quad \text{and} \quad c_{t_3,t_5,\text{Bob}} = (t_3, t_5, \{s_6\}, \{s_3\}).$$

As can be easily verified, both conditions now hold and we deal with a conflicting changeset. OntoWiki Mobile thus creates two versions, r_{t_5} and r_{t_6}, resulting from applying $c_{t_3,t_5,\text{Bob}}$ and $c_{t_3,t_6,\text{Alice}}$ to r_{t_3}, respectively.

4 User Interface

The OntoWiki Mobile user interface supports currently three different usage patterns: standard browsing along the taxonomic structures (e. g. class hierarchies) found in the knowledge base, faceted browsing for filtering instances based on property values as well as authoring of new information on-the-go.

4.1 Standard Browsing

Figure 3 shows the OntoWiki Mobile standard navigation user interface in different browsing states. In accordance with popular touch-oriented mobile software platforms, the user interface was based on lists so as to simplify navigating through interlinked resources. The first screenshot (Figure 3a) shows the list of all knowledge bases. The login button in the top-left corner allows to log in as a

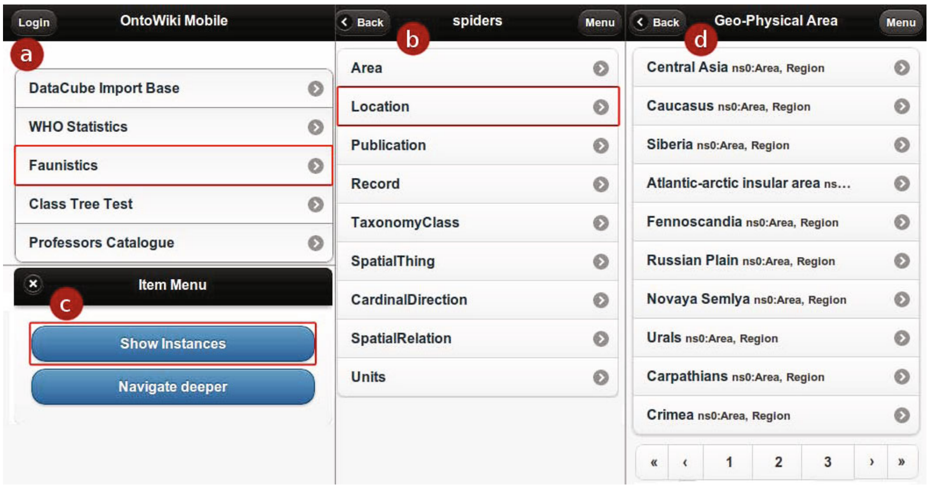


Fig. 3. OntoWiki Mobile standard browsing interface

registered user (e. g. to write to protected knowledge bases). After the user selects the knowledge base by tapping on it (tapped areas show as red squares), the top level class structure is displayed, as shown in second screenshot (Figure 3b). Once a particular class is chosen the user has to select (Figure 3c) whether he wants to see the list of instances for this class or navigate deeper in a class tree. Navigating through the class tree simply changes the classes list entries and refreshes the view. If the user chooses to view instances, a new list of instances from this class is presented (Figure 3d). After selecting a particular instance all properties are grouped by predicates and rendered in a list.

4.2 Faceted Browsing

Faceted browsing is a special way to navigate through instances in a specific knowledge base. It allows for simple and efficient filtering of the displayed instances list by applying available instance properties as filters. Faceted browsing can be used with any instance list in OntoWiki Mobile. As show in Figure 3d instance list view has a menu button in the upper-right corner. Using this button the user can access the instance list menu (Figure 4a), where he can execute a simple string search in current knowledge base or use the "Filters" button to access the faceted browsing feature. As shown in Figure 4b, the active filters list view displays all currently applied filters. By checking filters and pressing the "Delete" button at the bottom of the screen, the user can remove filters he does not like to apply to the list. To add a new filter the "Add" button in the upper-right corner of the screen can be used, which will display the list of all available filters (Figure 4c). Selecting one of the displayed filters will open the list of values for it (Figure 4d). Selecting values from the list shown will apply the new filter value restriction on the previously displayed instances list.

4.3 Authoring

Data authoring in OntoWiki Mobile is done using the RDFauthor [12] – a JavaScript-based system for RDF content authoring. As mentioned earlier,

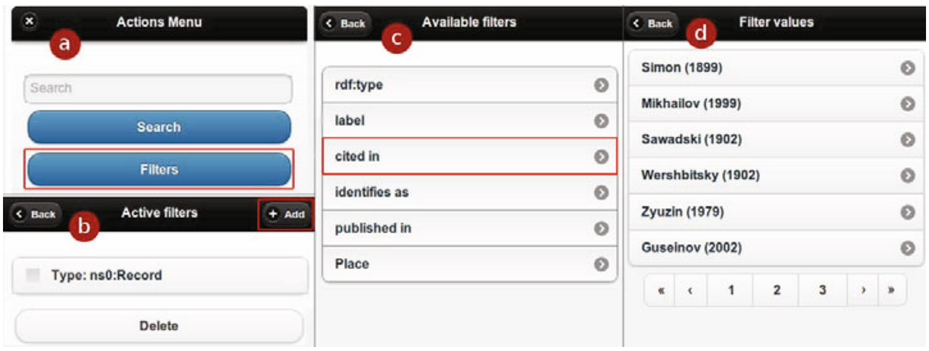
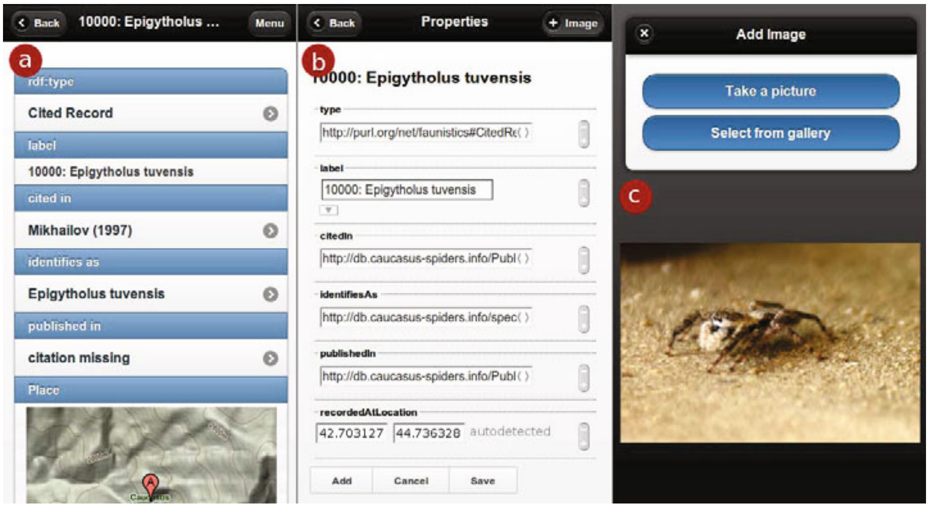


Fig. 4. OntoWiki Mobile faceted browsing interface



10000: Epigytholus ...

rdf:type
Cited Record
label
10000: Epigytholus tuvensis
cited in
Mikhailov (1997)
identifies as
Epigytholus tuvensis
published in
citation missing
Place

0000: Epigytholus tuvensis

type	http://purl.org/net/faunistics#CitedRe( )
label	10000: Epigytholus tuvensis
citedIn	http://db.caucasus-spiders.info/Publ( )
identifiesAs	http://db.caucasus-spiders.info/spec( )
publishedIn	http://db.caucasus-spiders.info/Publ( )
recordedAtLocation	42.703127 44.736328 autodetected

Fig. 5. OntoWiki Mobile authoring interface

instance properties are grouped by predicates and rendered as lists (Figure 5a). The rendering of properties and their values is based on the data type and ontology structure (see the display of a map for attached geo-coordinates). Figure 5b shows how an instance can be created or edited using forms, which are automatically created by RDFauthor based on the underlying ontology structure in the knowledge base (note auto-detected geographical coordinates for current location). Figure 5c shows an example of the interaction of OntoWiki mobile with the sensors of the mobile device in terms of accessing the integrated camera for adding a picture to an ontology instance.

5 Use Case and Evaluation

The development of OntoWiki Mobile was triggered by users aiming to gather data in field conditions. To simplify the data collection we created a mobile interface that allows users to enter data instances on mobile devices, such as mobile phones and tablets. In particular, there is a community of scientists who collect data about spiders in the Caucasus region [5] in a web portal[5]. The project consists of two major software parts:

– The portal backend, which is based on the semantic data wiki OntoWiki. Each arachnologist can login to this backend and use it for data entry, management and queries. The backend itself is a standard OntoWiki installation with some custom and some common vocabularies imported.

[5] http://db.caucasus-spiders.info

– The portal front-end, which itself is an extension of OntoWiki, generates a more visitor-friendly representation of the databases resources. The main focus for these visitors are species checklists, which give an overview of verified species of a given region.

To calculate these species checklists for a specific area, data from original finding spots (e. g. "I've found the species Pholcus phalangioides near Khashuri in Georgia in a cave.") as well as data from the literature (e. g. "Mkheidze (1964) published he found this species in Lentekhi as well.") is collected by the users. While the literature research is done at home using the desktop browser-based version of OntoWiki, the field data is gathered according to the following workflow:

1. A research team travels to the area and sets up traps at specific locations or specifically catches interesting individuals.
2. The finding spots are documented and the individual animals are associated with these finding spots (e. g. by signing a conservation container with the location ID).
3. The individuals are carried to the laboratory where they have to be identified. This is a challenging task and often individuals are sent to specialists for a specific genus or family of spiders.
4. Finally, the individual is identified as a certain species. This event either increases the finding spot counter for this species in a certain area or adds another species entry to the species checklist for this area. In the latter case, this (re-)discovery of the species in a certain area can be published.

In this workflow, only the second step is relevant for the evaluation of our work since step 3 and 4 are done with the standard wiki and the portal front-end. The goal of step 2 is to describe the finding spot where a specific animal was found in order to proof assumption about the habitat and living of a specific species. These finding spots are classified according to a nature-phenomenological system (e. g. a cave, field, . . .) and are allocated to a nearby populated place. Populated places are ordered and associated to a political and administrative system (e. g. counties, administrative regions, country). The database currently consists of 1060 populated places and locations as well as 191 areas. A complete finding spot documentation is then entered in the following steps (see N3 example in Figure 6):

– Instantiate a specific type of location (e. g. a cave, example line 8).
– Add geo-coordinates to this resource (taken automatically from the GPS subsystem, example line 13).
– Associate pictures with this resource (taken from the camera subsystem, example line 12).
– Associate a nearby populated place or an area by searching the local store for an existing one or by creating a new resource (example line 14).
– Add any other information either specific for the location type (e. g. height for glaciers), specific for the researcher (e. g. comments and tags) or specific for the research journey (e. g. internal location ID for the conservation containers, refer example line 9–11).

The result of a finding spot location documentation is shown in Figure 6.

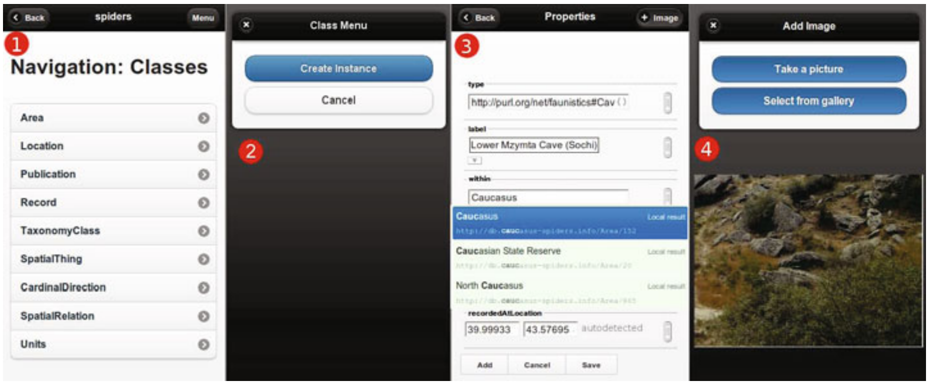
```
@prefix db: <http://db.caucasus-spiders.info/> .
@prefix faun: <http://purl.org/net/faunistics#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix geo: <http://www.w3.org/2003/01/geo/wgs84_pos#> .
@prefix dc: <http://purl.org/dc/elements/1.1/> .

<http://db.caucasus-spiders.info/Place/555> a faun:Cave ;
    rdfs:label "Lower Mzymta Cave (Sochi)";
    rdfs:comment "container 5";
    dc:creator "Stefan Otto"; dc:date "2010-07-21";
    foaf:depiction db:FotoXXX;
    geo:long "39.99933"; geo:lat "43.57695" ;
    faun:nearby db:Place19; faun:within db:Area439.
```

Fig. 6. Example finding spot documentation in N3

The data in this listing correspond to the screenshots shown in Figure 7.

After using the prototype for a few weeks on an Android developer phone, the following pro and con statements were obtained from the OntoWiki Mobile evaluation participants during interviews:

– Even without doing extensive data entry, the feature of associating images to existing resources was liked very much.
– There was a constant fear for data loss by breaking, misusing or loosing the mobile device. The added feature to export and import file backups from and to the application eased this. If there are more than one mobile device in the field, these backups can be additionally used to approve and inspect the data with a second pair of eyes.
– Obtaining GPS data from the mobile phone is nice but slightly inaccurate since the internal GPS systems of mobile phones are not as good as dedicated devices e. g. for hiking. Auto-completion of these values is a nice feature but users need to be able to correct them or, even better, receive them from another device and overwrite the existing values.
– The user experience strongly depends on the given CPU of the mobile device. Users running a mobile device with 500Mhz (HTC Hero) complained about the slowly responding user interface. In comparison to that, users with a 1000Mhz device (Samsung Galaxy S) reported a fast and reliable interface. In addition to the CPU, the version of the hosting Android operating system and esp. the used browser version strongly affects the user experience since newer Android versions also ship a new browser with a faster JavaScript execution engine.



type
http://purl.org/net/faunistics#Cav ()

label
Lower Mzymta Cave (Sochi)

within
Caucasus

Caucasus Local result
http://db.caucasus-spiders.info/Area/152

Caucasian State Reserve Local result
http://db.caucasus-spiders.info/Area/20

North Caucasus Local result
http://db.caucasus-spiders.info/Area/965

recordedAtLocation
39.99933 43.57695 autodetected

Fig. 7. Screenshots illustrating the workflow for creating a new finding spot according to the listing in Figure 6. From left to right: (1.) Searching or browsing for the class which needs to be instantiated. (2.) Initialization of a new resource from this class; all properties which are offered, are used in other instances of this class; GPS data is automatically requested and pre-filled by the phone. (3.) Entering literal data as well as linking to other resources. (4.) Assignment of existing images from the phone's image library.

6 Related Work

Related work can be roughly divided into the categories mobile semantic applications, strategies for replication, reconciliation in mobile usage environments.

6.1 Mobile Semantic Applications

The application of Semantic Web technologies on mobile devices is not new – one of the earlier works dates back to 2003 [4]. However, is was not pursued very actively due to the large number of mobile device limitations common in that era. In the light of increasing processing power, data connectivity and flexibility of modern mobile devices, the use of mobile Semantic Web technologies is becoming more feasible. There are a few publications which review the state of the mobile Semantic Web and the possibilities thereof to improve mobile Web (e. g. [3]). Also, there are publications on more complex topics like Semantic Web system for mobile devices named *SmartWeb* [11] which uses semantic technologies in combination with device specific input capabilities (e. g. voice input) to enhance mobile web services. On the frontend side there are semantic mobile applications like *DBpedia Mobile*[1] or *mSpace Mobile* [14] that implement or utilize the Semantic Web technologies directly on mobile devices. However, these frontend applications focus on very specific use cases – information about points of interest in the case of DBpedia Mobile and information for university students in the case of mSpace. OntoWiki Mobile on the other hand is a generic, application domain agnostic tool, which can be utilized in a wide range of very different usage scenarios ranging from instance acquisition to browsing of semantic data on the go.

6.2 Strategies for Replication

One of the most anticipated topics in distributed semantic data bases is data replication. Though, the topic of semantic data replication and conflict resolution in semantic data bases is not new, most of the existing approaches developed for desktop application rely heavily on client resources. On the other hand, topic of data replication in distributed databases on mobile devices is highlighted very thoroughly[6]. Semantic data replication for mobile devices adopts some of the approaches for generic mobile databases. There are generic approaches like change detection using a *Version Log* [7]. Also, there are specific approaches allowing to create an ontology versioning system for a particular RDF-based ontology language like *SemVersion* [13]. There are a large number of publications on data versioning and replication in the distributed database area, but most of these approaches require substantial computational power from client devices (which is not reasonably applicable in the case of mobile devices).

6.3 Reconciliation in Mobile Usage Environments

There is already a significant number of publications about data replication and versioning in the mobile Semantic Web environment. There are currently several trends in existing approaches:

– Complex client-side replication engines (like *MobiSem Replication and Versioning framework* [10]) that provide functionality to make relevant data available on the mobile device and to synchronize changes when connectivity is recovered.
– Enrichment of graphs with additional metadata (like *Triple Bitmaps* [9]) to simplify replication, merging and conflict resolution.

However, the implementation usually requires full-fledged RDF storage on the client-device with additional layers of versioning functionality. Adding such a additional layer is not always possible or difficult to implement for the large variety of existing mobile target platforms. OntoWiki Mobile on the other hand does not require any client-side technology beyond support for HTML5.

7 Conclusions

As the penetration of mobile devices able to access and interact with the Web can be expected to dramatically increase within the next years, the Semantic Web can ultimately only be successful if the use of semantic technologies on mobile devices is fully supported. With OntoWiki Mobile we tackled one particular but crucial aspect – the provisioning of a comprehensive knowledge management tool for mobile use. It employs the new HTML5 application cache functionality to support offline work and has advanced conflict resolution features built-in. OntoWiki Mobile demonstrates that a comprehensive semantic collaboration platform is possible to implement for mobile devices with minimal requirements based on recent Web standards (in particular HTML5). Although OntoWiki

Mobile was already used in a specific use-case by a number of non-IT, domain expert users more effort is required to evaluate the tool in a wider range of application scenarios. Due to its general purpose architecture OntoWiki Mobile is particularly suited to support the long tail of domain-specific mobile applications, for which the development of individual tools would not be (economically) feasible.

Future work will focus on representation of provenance and use of the mobile device's sensors for context-aware knowledge base exploration. With regard to the replication we plan to develop a rule-based approach for the selection of knowledge base parts to replicate on the mobile device. The approach will take mobile context information (such as the time, location) as well as usage patterns (e. g. browsing history) and manually supplied user preferences into account.

Acknowledgments

We would like to thank our colleagues from AKSW research group for their helpful comments and inspiring discussions during the development of this approach. This work was supported by a grant from the European Union's 7th Framework Programme provided for the project LOD2 (GA no. 257943).

References

1. Becker, C., Bizer, C.: Exploring the Geospatial Semantic Web with DBpedia Mobile. J. Web Sem. 7(4), 278–286 (2009)
2. Heino, N., Dietzold, S., Martin, M., Auer, S.: Developing Semantic Web Applications with the Ontowiki Framework. In: Pellegrini, T., Auer, S., Tochtermann, K., Schaffert, S. (eds.) Networked Knowledge - Networked Media. SCI, vol. 221, pp. 61–77. Springer, Heidelberg (2009)
3. Lassila, O.: Using the Semantic Web in Mobile and Ubiquitous Computing. In: Proceedings of the 1st IFIP WG12.5 Working Conference on Industrial Applications of Semantic Web, Jyväskylä, Finland, August 25-27 (2008)
4. Lassila, O., Adler, M.: Semantic Gadgets: Ubiquitous Computing Meets the Semantic Web. In: Fensel, D., Hendler, J.A., Lieberman, H., Wahlster, W. (eds.) Spinning the Semantic Web, pp. 363–376. MIT Press, Cambridge (2003)
5. Otto, S., Dietzold, S.: Caucasian Spiders – A faunistic Database on the spiders of the Caucasus. Newsl. Brit. Arachn. Soc. 108, 14 (2007), http://caucasus-spiders.info
6. Padmanabhan, P., Gruenwald, L., Vallur, A., Atiquzzaman, M.: A survey of data replication techniques for mobile ad hoc network databases. The VLDB Journal 17, 1143–1164 (2008)
7. Plessers, P., De Troyer, O.: Ontology Change Detection Using a Version Log. In: Gil, Y., Motta, E., Benjamins, V.R., Musen, M.A. (eds.) ISWC 2005. LNCS, vol. 3729, pp. 578–592. Springer, Heidelberg (2005)
8. Rieß, C., Heino, N., Tramp, S., Auer, S.: EvoPat – Pattern-Based Evolution and Refactoring of RDF Knowledge Bases. In: Patel-Schneider, P.F., Pan, Y., Hitzler, P., Mika, P., Zhang, L., Pan, J.Z., Horrocks, I., Glimm, B. (eds.) ISWC 2010, Part I. LNCS, vol. 6496, pp. 647–662. Springer, Heidelberg (2010)

9. 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. LNCS, vol. 6088, pp. 31–45. Springer, Heidelberg (2010)
10. Schandl, B., Zander, S.: A Framework for Adaptive RDF Graph Replication for Mobile Semantic Web Applications. In: Joint Workshop on Advanced Technologies and Techniques for Enterprise Information Systems (Session on Managing Data with Mobile Devices), Milan, Italy, pp. 154–163. INSTICC Press (May 2009)
11. Sonntag, D., Engel, R., Herzog, G., Pfalzgraf, A., Pfleger, N., Romanelli, M., Reithinger, N.: SmartWeb handheld — multimodal interaction with ontological knowledge bases and semantic web services. In: Huang, T.S., Nijholt, A., Pantic, M., Pentland, A. (eds.) ICMI/IJCAI Workshops 2007. LNCS (LNAI), vol. 4451, pp. 272–295. Springer, Heidelberg (2007)
12. Tramp, S., Heino, N., Auer, S., Frischmuth, P.: RDFauthor: Employing rDFa for collaborative knowledge engineering. In: Cimiano, P., Pinto, H.S. (eds.) EKAW 2010. LNCS, vol. 6317, pp. 90–104. Springer, Heidelberg (2010)
13. Völkel, M., Groza, T.: SemVersion: RDF-based ontology versioning system. In: Proceedings of the IADIS International Conference WWW/Internet 2006, ICWI (2006)
14. Wilson, M., Russell, A., Smith, D.A., Owens, A., Schraefel, M.C.: mSpace Mobile: A Mobile Application for the Semantic Web. In: Proc. of the ISWC 2005 Workshop on End User Semantic Web Interaction. CEUR Workshop Proceedings, vol. 172, CEUR-WS.org (2005)

Weaving a Distributed, Semantic Social Network for Mobile Users

Sebastian Tramp, Philipp Frischmuth, Natanael Arndt, Timofey Ermilov, and Sören Auer

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

Abstract. Smartphones, which contain a large number of sensors and integrated devices, are becoming increasingly powerful and fully featured computing platforms in our pockets. For many people they already replace the computer as their window to the Internet, to the Web as well as to social networks. Hence, the management and presentation of information about contacts, social relationships and associated information is one of the main requirements and features of today's smartphones. The problem is currently solved only for centralized proprietary platforms (such as Google mail, contacts & calendar) as well as data-silo-like social networks (e.g. Facebook). Within the Semantic Web initiative standards and best-practices for social, Semantic Web applications such as FOAF emerged. However, there is no comprehensive strategy, how these technologies can be used efficiently in a mobile environment. In this paper we present the architecture as well as the implementation of a mobile Social Semantic Web framework, which weaves a distributed social network based on semantic technologies.

1 Introduction

Smartphones, which contain a large number of sensors and integrated devices, are becoming increasingly powerful and fully featured computing platforms in our pockets. For many people they already replace the computer as their window to the Internet, to the Web as well as to social networks. Hence, the management and presentation of information about contacts, social relationships and associated information is one of the main requirements and features of today's smartphones.

The problem is currently solved solely for *centralized* proprietary platforms (such as Google mail, contacts & calendar) as well as data-silo-like social networks (e.g. Facebook). As a result of this data centralization, users' data is taken out of their hands, they have to accept the predetermined privacy and data security regulations; users are dependent of the infrastructure of a single provider, they experience a lock-in effect, since long-term collected profile and relationship information cannot be easily transferred. Increasingly, many people argue that

© Springer-Verlag Berlin Heidelberg 2011

social networks should be evolving. That is, they should allow users to control what to enter and to keep a control over their own data. Also, the users should be able to host the data on an infrastructure, which is under their direct control, the same way as they host their own website [3].

A possibility to overcome these problems and to give the control over their data back to the users is the realization of a truly *distributed* social network. Initial approaches for realizing a distributed social network appeared with *GNU social* and more recently *Diaspora* (cf. Section 6). However, we argue that a distributed social network should be also based on semantic resource descriptions and de-referenceability so as to ensure versatility, reusability and openness in order to accommodate unforeseen usage scenarios.

Within the Semantic Web initiative already a number of standards and best-practices for social, Semantic Web applications such as *FOAF*, *WebID* and *Semantic Pingback* emerged. However, there is no comprehensive strategy, how these technologies can (a) be combined in order to weave a truly open and distributed social network on the Web and (b) be used efficiently in a mobile environment. Also, the use of a distributed, social semantic network should be as *simple* as the use of the currently widely used centralized social networks (if not even simpler). In this paper we present the general strategy for weaving a distributed social semantic network based on the above mentioned standards and best-practices. In order to foster its adoption we developed an implementation for the Android platform, which seamlessly integrates into the commonly used interfaces for contact and profile management on mobile devices.

After briefly reviewing some use cases and requirements for a mobile, semantic social network application (in Section 2), we make in particular the following contributions:

– We outline a strategy to combine current bits and pieces of the Semantic Web technology realm in order to realize a distributed, semantic social network (Section 3),
– We develop an architecture for making mobile devices endpoints for the Social Semantic Web (Section 3),
– A comprehensive implementation of the architecture was performed for the Android platform (Section 4 and 5),

Furthermore, our paper contains an overview on related work in Section 6 and concludes in Section 7 with a discussion and outlook on future work.

2 Mobile Use Cases and Requirements

Before describing the overall strategy, the technical architecture and our implementation we want to briefly outline in this section the key requirements, which guided our work. These requirements are common sense in the context of social networks and are not newly coined by us. Unfortunately most of them are not achieved in the context of semantics enabled and distributed social networks, so we describe them especially from this point of view.

Make new friends. Adding new contacts to our social network is the precondition in order to gather useful information from this network. Maintaining our social network directly from your mobile phones means that we are able to instantly connect with new contacts (e.g. on conferences or parties). In the context of a distributed social network, this use-case also includes the employment of semantic search engines to acquire the WebID of a new contact based on parts of its information (typically the contacts name). In order to shorten the overall effort for adding new contacts, functionality for scanning and decoding a contacts business cards QR code[1] are also included in this use-case.

Be in sync with your social network. Once our social network is woven and social connections are established, we want to be able to gather information from this network. For a distributed social network this means, that a combination of push and pull communications is needed to be as timely updated as needed and as fast synced as possible. Especially this use-case is bound to a bunch of access control requirements[2]. where people want to permit and deny access to specific information in fine grained shades and based on groups, live contexts and individuals.

Annotate contacts profiles. It should be possible to annotate profiles of contacts freely, e.g. with updated information, contact group categorizations (e.g. friends, family, co-workers). These annotations should be handled in the same way as the original data from the friend's WebID except that this data is not updated with the WebID but persists as an annotation. One additional feature request in this use-case is to share these annotations across ones personal devices on the web, e.g. by pushing them to a triple store which is attached to ones WebID.

General requirements. The development of the Mobile Social Semantic Web Client was driven by a few general requirements which derived from our own experience with mobile phones and FOAF-based WebIDs:

– *Be as decent as possible:* Today's FOAF-based social networks are mostly driven by uploaded RDF files. In order to support such low end profiles, there should be no other required feature on a WebID than the availability as Linked Data[3]. All other features (FOAF+SSL, Semantic Pingback, subscription service) should be handled as optional and our client should require as little infrastructure as possible.

[1] QR codes are two-dimensional barcodes which can encode URIs as well as other information. They are especially famous in Japan, but their popularity grows more and more worldwide since mobile applications for decoding them with a standard camera can be used on a wide range of devices.

[2] A typical requirement: Disallow access to my mobile number except for friends and family members.

[3] In the meaning of Linked RDF Data defined at `http://www.w3.org/DesignIssues/LinkedData`

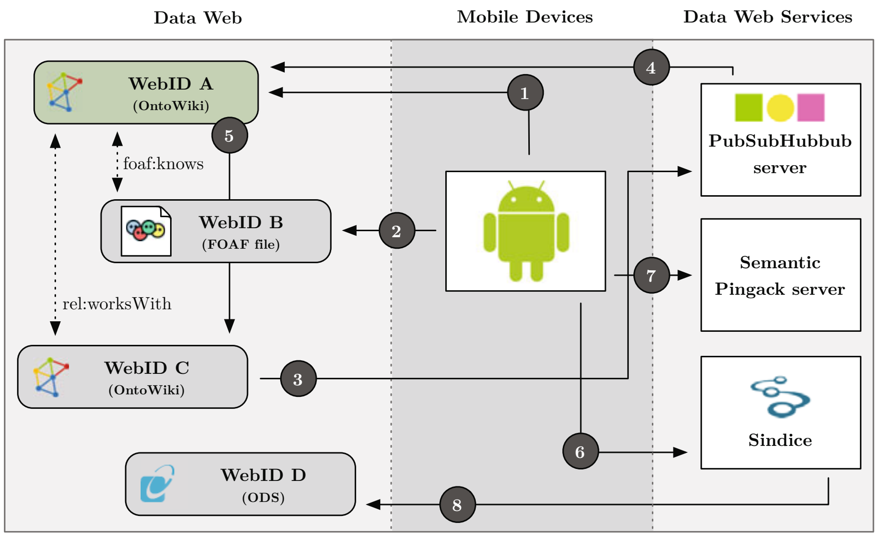


Fig. 1. Architecture of a distributed, semantic social network: (1) A mobile user may retrieve updates from his social network via his WebID provider, e.g. from OntoWiki. (2) He may also fetch updates directly from the sources of the connected WebIDs. (3) A WebID provider can notify a subscription service, e.g. a PubSubHubbub server, about changes. (4) The subscription service notifies all subscribers. (5) As a result of a subscription notification, another node can update its data. (6) A mobile user can search for a new WebID by using a semantic search engine, e.g. Sindice. (7) To connect to a new WebID he sends a Pingback request which (8) notifies of the resource owner.

- *Be as transparent as possible:* Mobile user interfaces are built for efficiency and daily use. People become accustomed with them and any changes in the daily work flow of using information from the social network will annoy them. The client we had in mind should work mostly invisible from the user, which means it should be well integrated into the hosting mobile operating system.
- *Be as flexible as possible:* This is especially needed in an environment where vocabularies are not yet standardized and are subject to changes and extensions. Our solution should be flexible in the sense that we do not want built-in rules on how to deal with specific attributes or relations.

Based on these preliminaries as well as based on the Social Semantic Web state of the art, we describe an architecture of a distributed social semantic network in the next section.

3 Architecture of a Distributed Semantic Social Network

In this section we describe the main ingredients for a distributed, semantic social network as well as their interplay. The overall architecture is depicted in

Figure [1]. The semantic representation of personal information is facilitated by WebID. FOAF+SSL allow the use of a WebID for authentication and access control purposes. Semantic Pingback facilitates the first contact between users of the social network and subscription services allow obtaining specific information from people in ones social network as near-instant notifications.

WebID. WebID [16] is a best-practice recently conceived in order to simplify the creation of a digital ID for end users. Since its focus lies on simplicity, the requirements for a WebID are minimal. In essence, a WebID is an de-referenceable RDF document (including RDFa) describing its owner[4]. That is, a WebID contains RDF triples, which have the IRI identifying the owner as subject. The description of the owner can be performed in any (mix of) suitable vocabularies, but FOAF [4] emerged as the 'industry standard' for that purpose. An example WebID comprising some personal information (lines 8-12) and two `rel:worksWith`[5] links to co-workers (lines 6-7) is shown in Listing [1].

```
1  @prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
2  @prefix foaf: <http://xmlns.com/foaf/0.1/> .
3  @prefix rel: <http://purl.org/vocab/relationship/> .
4  <http://philipp.frischmuth24.de/id/me> a foaf:Person;
5    rdfs:comment "This is my public profile only, more
         information available with FOAF+SSL";
6    rel:worksWith <http://sebastian.tramp.name>,
7      <http://www.informatik.uni-leipzig.de/~auer/foaf.rdf#me>;
8    foaf:depiction <http://img.frischmuth24.de/people/me.jpg>;
9    foaf:firstName "Philipp"; foaf:surname "Frischmuth";
10   foaf:mbox <mailto:frischmuth@informatik.uni-leipzig.de>;
11   foaf:phone <tel:+49-341-97-32368>;
12   foaf:workInfoHomepage <http://bis.informatik.uni-leipzig.de
         /PhilippFrischmuth>.
```

Listing 1. A minimal WebID with personal information and two *worksWith* relations to other WebIDs

FOAF+SSL. The more technical FOAF+SSL best-practice [17] aims to incorporate authentication functionality into the WebID concept. The main idea is to link an SSL client certificate to a WebID, thus allowing the owner of the FOAF+SSL enabled WebID to authenticate herself at 3rd party websites. Another goal of FOAF+SSL is to provide access control functionality for a social network shaped by WebIDs in order to allow access to different kinds of information for different groups of contacts (e.g. as presented with dgFOAF [14]). An example of a FOAF+SSL WebID extension is shown in Listing [2]. This WebID

[4] The usage of an IRI with a fragment identifier allows the indirect identification of a WebID by reference to the (FOAF) profile document.

[5] Taken from *RELATIONSHIP: A vocabulary for describing relationships between people* at `http://purl.org/vocab/relationship`

now contains a description of an RSA public key (line 15), which is associated to the WebID by using the `cert:identity` property from the W3C certificates and crypto ontology (line 19).

```
13 @prefix rsa: <http://www.w3.org/ns/auth/rsa#> .
14 @prefix cert: <http://www.w3.org/ns/auth/cert#"> .
15 [] a rsa:RSAPublicKey;
16   rdfs:comment "used from my smartphone ...";
17   cert:identity <http://philipp.frischmuth24.de/id/me>;
18   rsa:modulus          [ cert:hex "C41199E ... 5AB5" ];
19   rsa:public_exponent  [ cert:decimal "65537" ] .
```

Listing 2. Extension of the minimal WebID from Listing 1: Description of an RSA public key, which is associated to the WebID by using the `cert:identity` property from the W3C certificates and crypto ontology

Semantic Pingback. The purpose of Semantic Pingback [18] in the context of a distributed social network is to facilitate the first contact between different people using the network. The approach is based on an extension of the well-known Pingback technology [8], which is one of the technological cornerstones of the overwhelming success of the blogosphere in the Social Web. The Semantic Pingback mechanism enables bi-directional links between WebIDs, RDF resources as well as weblogs and websites in general. It facilitates contact/author/user notifications in case a link has been newly established. It is based on the advertising of a lightweight RPC service, in the RDF document, HTTP or HTML header of a certain Web resource, which should be called as soon as a (typed RDF) link to that resource is established. The Semantic Pingback mechanism enables people but also authors of RDF content, a weblog entry or an article in general to obtain immediate feedback, when other people establish a reference to them or their work, thus *facilitating social interactions*. It also allows to automatically publish backlinks from the original WebID (or other content) to comments or references of the WebID (or other content) elsewhere on the Web, thus *facilitating timeliness and coherence* of the Social Web. As a result, the distributed network of WebID profiles, RDF resources and social websites using the Semantic Pingback mechanism can be much tighter and timelier interlinked than conventional websites, thus rendering a network effect, which is one of the major success factors of the Social Web. Semantic Pingback is completely downwards compatible with the conventional Pingback implementations, thus allowing the seamless connection and interlinking of resources on the Social Web with resources on the Data Web. An extension of our example profile with Semantic Pingback functionality making use of an external Semantic Pingback service is shown in Listing 3.

Subscription Service. The purpose of a WebID subscription service is to establish a publish/subscribe communication model to provide near-instant notifications of contact updates. The main idea here is to extend a WebID with a link to

```
@prefix ping: <http://purl.org/net/pingback/> .
<http://philipp.frischmuth24.de/id/me> ping:to <http://
    pingback.aksw.org>.
```

Listing 3. Extension of the minimal WebID from Listing 1: Assignment of an external Semantic Pingback service which can be used to ping this specific resource

a *PubSubHubbub* service[6] where any contact can subscribe to the WebIDs updates. Although such a behavior is described for SPARQL results in [12], there is currently no standardized solution for publishing RDF change sets through PubSubHubbub as well as for saving the incoming changes from all friends of a user in some kind of cache or proxy while the mobile device (the subscriber) is not online. As a consequence, our implementation (as described in the next section) does not yet support a full-fledged update subscription. As a fallback, updates are currently polled from the related WebIDs. This increases network bandwidth usage and might lead in some cases to slower user interfaces due to network latency. Please refer to Section 7 for a description of possible future work in this direction.

4 Implementation of a Mobile Interface

After describing the architecture of a distributed, semantic social network we now present our implementation of a mobile interface for this network.

4.1 Android System Integration

Figure 2 depicts the mobile social Semantic Web client consisting of two application frameworks, which are built on top of the Android runtime and a number of libraries. In particular, *androjena*[7] is one of those libraries, which itself is a partial port of the popular Jena framework[8] to the Android platform. Both frameworks provided by the client share the feature that they are accessible through content providers. The Mobile Semantic Web middleware (MSW) is responsible for importing Linked Data resources (in particular via FOAF+SSL) and persisting that data. It operates on triple level and provides access to the various triple stores through a content provider called `TripleProvider`. Each resource is stored separately, since named graphs are currently not supported. The Mobile Social Semantic Web middleware (MSSW) queries the triple data provided by MSW and transforms that data into a format that is more appropriate for social applications. It propagates two content providers, one that integrates well with the layout of contact information on Android phones (`ContactProvider`) and one that is suitable for FOAF based applications (`FoafProvider`).

[6] PubSubHubbub is an open, server-to-server web-hook-based publish/subscribe protocol realized as an extension to Atom: `http://code.google.com/p/pubsubhubbub/`

[7] `http://code.google.com/p/androjena/`

[8] `http://jena.sourceforge.net/`

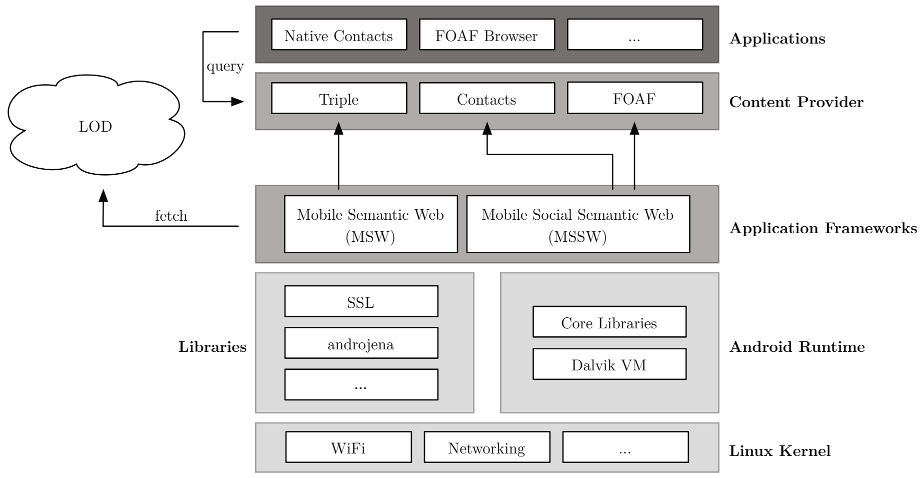


Fig. 2. Android Integration Layer Cake

4.2 Model Management

Since WebIDs are Linked Data enabled, they usually return data describing that resource. This circumstance makes it feasible to store a graph (referred to as a model here) for each WebID, since the redundancy between models is expected to be marginal. In reality MSW keeps more than one model per WebID for different purposes. On the mobile phones' SD-card we keep these models in the following subdirectories:

- `web` – This folder contains exact copies of the documents retrieved from the Web.
- `inf` – Models stored in this folder contain all entailed triples (more on this in Section 4.3).
- `local` – The user can annotate all WebIDs with personal information, which will be stored in this folder.

We decided to store all data as RDF files on a swappable SD-card, since we expect the following user benefits:

- Because SD-cards can be exchanged, the data is portable and can be reused on another phone or device. This makes the whole system more fail-proof.
- Most modern computers can handle SD-cards and hence data can be easily backed up.
- Other applications on the Android phone running the mobile Semantic Web client can access and modify the data stored on the card. Thus they can further annotate the information and the client can again take advantage of such annotations.

4.3 Rules and Data Processing

One of our initial requirements from Section 2 is flexibility in the sense that specific vocabulary resources should not be encoded in the source code of the WebID provider. In order to achieve this requirement, we decided to encode as much data processing as possible in terms of user extensible rules. Since we employ the *androjena framework*, we were able to use the included Jena rules engine as well. All rules processed by this rule-based reasoner are defined as lists of body terms (premises), lists of head terms (conclusions) and optional names[9].

```
 1  @prefix foaf: <http://xmlns.com/foaf/0.1/>.
 2  @prefix android: <http://ns.aksw.org/Android/>.
 3  @prefix acontacts: <http://ns.aksw.org/Android/
        ContactsContract.CommonDataKinds.>.
 4  @prefix im: <http://ns.aksw.org/Android/ContactsContract.
        CommonDataKinds.Im.>.
 5
 6  [jabber:
 7    (?s foaf:jabberID ?o), makeTemp(?d) ->
 8      (?s android:hasData ?d),
 9      (?d rdf:type acontacts:Im),
10      (?d im:DATA ?o),
11      (?d im:TYPE im:TYPE_HOME),
12      (?d im:PROTOCOL im:PROTOCOL_JABBER)
13  ]
```

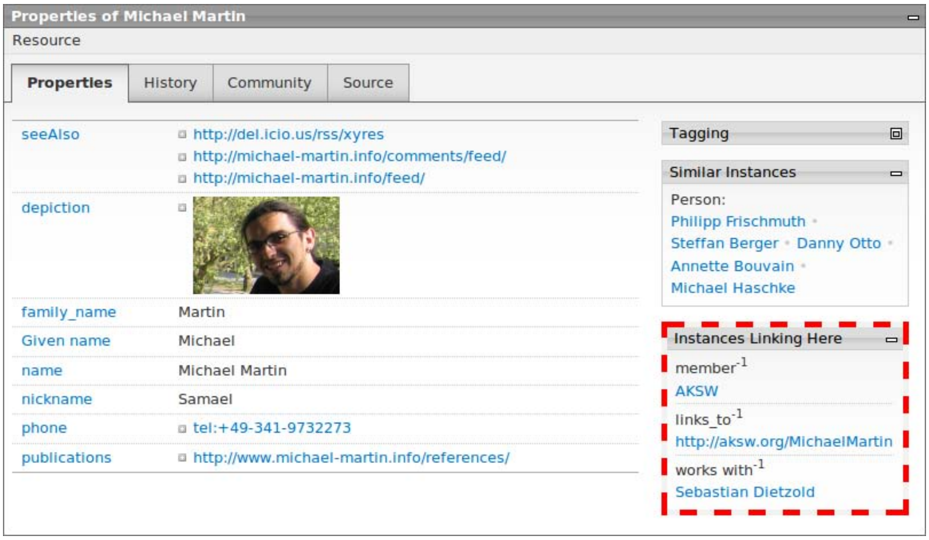
Listing 4. Example transformation rule: If a `foaf:jabberID` is present with a WebID (line 7), then a new blank node of RDF type `acontacts:Im` is created (line 7), which is of Android IM type `HOME` (line 11) and which gets an IM protocol as well as the IM identifier (line 12 and 10)

Since we also did not want our implementation to depend on the FOAF vocabulary (alternative solutions include RDF vCards [7]), we decided to create a native Android system vocabulary which represents the Android contacts database defined by the Android API. This vocabulary is deeply integrated into the Android system since it re-uses class and attribute names from the Android API and represents them as OWL class and datatype properties[10].

Based on this vocabulary, the given rules transform the downloaded WebID statements into Android-specific structures which are well suited for a straightforward import into the contacts provider. These structures are very flat and

[9] `http://jena.sourceforge.net/inference/#RULEsyntax`

[10] An example class name is `ContactsContract.CommonDataKinds.StructuredName`, which is represented in the vocabulary as an OWL class with the URI `http://ns.aksw.org/Android/ContactsContract.CommonDataKinds.StructuredName`. We published the vocabulary at `http://ns.aksw.org/Android/`. Please have a look at the Android API reference as well (`http://developer.android.com/`).


Properties of Michael Martin

Resource

Properties | History | Community | Source

seeAlso	http://del.icio.us/rss/xyres http://michael-martin.info/comments/feed/ http://michael-martin.info/feed/
depiction	
family_name	Martin
Given name	Michael
name	Michael Martin
nickname	Samael
phone	tel:+49-341-9732273
publications	http://www.michael-martin.info/references/

Tagging

Similar Instances

Person:
Philipp Frischmuth • Steffan Berger • Danny Otto • Annette Bouvain • Michael Haschke

Instances Linking Here

member^{-1}
AKSW

links_to^{-1}
http://aksw.org/MichaelMartin

works with^{-1}
Sebastian Dietzold

Fig. 3. Visualization of a WebID in OntoWiki: incoming backlinks (via Semantic Pingback) are rendered in the "Instances Linking Here" side box

relate different Android data objects (e.g. email, photo, structured name etc.) via a `hasData` property to a WebID. An example rule which creates an instant messaging account for the contact is presented in Listing 4.

After applying the given set of rules, the application post-processes the generated data in order to apply other constraints which we could not achieve with Jena rules alone. At the moment all `mailto:` and `tel:` resources are transformed to literal values, which is required for instantiating the corresponding Java class. In addition we download, resize and base64-encode all linked images. After that, the application goes through the generated data resources and imports them one by one.

4.4 OntoWiki

The mobile Semantic Web client supports arbitrary WebIDs, even those backed by plain RDF files. Nevertheless, some features require special support on the server-side. For our semantic data wiki OntoWiki [1] we implemented all functionalities required for a complete distributed Social Web experience. Any user can setup his own OntoWiki instance, which will then provide him with an enhanced WebID.

If configured properly a user can create a self-signed certificate with very little effort. Such a certificate contains the generated WebID as a Subject Alternative Name (SAN) and is directly imported into supported Web browsers[11]. From the browser the certificate can be exported in *PKCS12* format and stored on a

[11] A list of supported browsers is available at `http://esw.w3.org/Foaf+ssl/Clients`.

SD-card used by a mobile phone running the client. Since OntoWiki supports FOAF+SSL authentication, a user can split his data in publicly visible information and such, that is only accessible by people which have a certain relationship with the user (e.g. a `foaf:knows` relation).

Semantic Pingback is another technology supported by OntoWiki. Thus an arbitrary user can add a relationship to an OntoWiki backed WebID and as a result the WebID owner will be notified, enabling the user to take further actions (see Figure 3). In the use case of the mobile Semantic Web client this is especially useful for a first contact between users. In typical social network applications this step would be the “Add as a friend" step. In a distributed scenario, however, if one states that she is a friend of someone else, she would allow that person to view the data dedicated to be displayed by friends only. If both endpoints add that relation on their respective side, they can see each other’s private data and thus are considered friends (in the Social Web sense). The Social Web has a very dynamic nature and information is changed frequently or new data is added. Hence, editing functionality is another important aspect and OntoWiki supports editing via SPARQL/Update.

5 User Perspective

The Mobile Semantic Social Web client implementation consists of two software packages - the *Android Semantic Web Core library* containing the triple store and the *WebID content provider for Android*. Both are available on the *Android Market* since August 2010 (cf. screenshot A in Figure 4). According to the market statistics, they were downloaded overall more than 400 times and are currently installed on more than 100 devices.

Once installed few initial configuration options have to be supplied. Screenshot B in Figure 4 shows the accounts and sync settings configuration menu, which allows a user to associate his WebID with his profile on the smartphone (the same way as adding an LDAP or Exchange account) and to configure synchronization intervals. Screenshot C shows an actual WebID with the last synchronization date and the option to trigger the synchronization manually.

After the user associated his profile with his WebID, information from linked WebIDs of the users contacts are synchronized regularly and the information are made available via the Android content provider to all applications on the device. During the import of the WebID contacts, they are merged based on the assumption of unique names. Independent of this automatic merge, the user can split and merge contacts manually in the edit view of these contacts. Screenshot D shows the standard Android contact application, where our WebID content provider seamlessly integrates information obtained from WebIDs. Information obtained from WebIDs is not editable, since it is retrieved from the authoritative sources, i.e. the WebIDs of the respective contacts.

Screenshot E shows the FOAF browser, allowing people to add contacts or to browse the contacts of their friends. In order to facilitate the process of connecting with new contacts the Android implementation also allows to scan QR-codes of WebIDs (e.g. from business cards) and to search for WebIDs using Sindice.

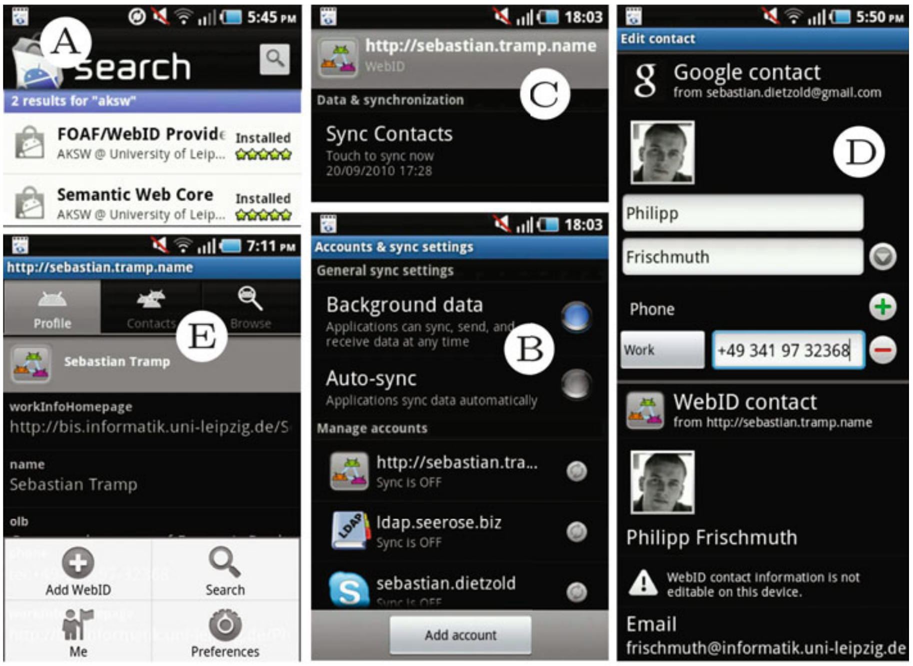


Fig. 4. Screenshots of the Mobile Social Semantic Web Client, the FOAF Browser and the Android components which integrate the WebID account: (A) The client as well as the triple store can be found in the official Google application market. (B) After installation, users can add a WebID account the same way they add an LDAP or Exchange account. (C) The account can be synchronized on request or automatically. (D) A contacts profile page merges the data from all given accounts. (E) By using the FOAF browser, people can add contacts or browse the contacts of their friends.

6 Related Work

Related work can be roughly divided into semantics-based (but centralized) social network services, distributed social network projects, mobile Semantic Web projects and mobile social network clients. A comprehensive overview is contained in the final report of the W3C Social Web Incubator Group [6]. In the sequel we present some related approaches along the four dimensions in more detail.

Semantics-based (but centralized) social network services. Evri[12] is a centralized social network based on RDF and mainly used for collaborative information storage. Evri also maintains mobile applications for iOS and Android to give their users access to these information.

[12] `http://www.evri.com` (formally know as Twine).

Distributed social network. The idea of distributed social networks appeared quickly after social networks became popular. A distributed social network traditionally refers to an Internet social network service that is decentralized and distributed across different providers, with emphasis on portability and interoperability. The most prominent representatives are Diaspora[13], NoseRub[14] and GNU social[15]. Although some of these networks make use of vocabularies (e.g. FOAF in the case of GNU Social) or certain elements of semantic technologies, their use of Semantic Web technologies and best-practices is rather limited.

Mobile Semantic Web applications. The application of Semantic Web technologies on mobile devices is not new - one of the earlier works dates back to 2003 [10]. However, this research area was not pursued very actively due to the large number of mobile device limitations common in that era. In the light of increasing processing power and data connectivity of modern mobile devices, the use of mobile Semantic Web technologies is becoming more feasible. There are a few publications which review the state of the mobile Semantic Web and the possibilities thereof to improve mobile Web (e.g. [9]). Also, there are publications on more complex systems, such as *SmartWeb* [15] which uses semantic technologies to enhance the backends of mobile web service. On the frontend side there are semantic mobile applications like *DBpedia Mobile*[2] or *mSpace Mobile* [19] that implement or utilize Semantic Web technologies directly on mobile devices. However, these frontend applications focus on very specific use cases - information about points of interest in the case of DBpedia Mobile and information for university students in the case of mSpace. Finally, there are publications which describe proof of concept applications as well as algorithms for consuming and replicating Linked Data and RDF in general [5,11,13].

Mobile social networking clients. All major social networking services (such as Facebook, Myspace, LinkedIn etc.[16]) have meanwhile clients for different mobile platforms, which are more or less integrated with the mobile phone platform itself. In addition to this, the Android market place lists more than 3500 applications in the category social networks with our implementation being one of them. However, up to our knowledge our MSSW client is the first to consequently employ W3C standards as well as Social Semantic Web best practices with regard to all aspects of data representation and integration.

7 Conclusion and Future Work

We see the work described in this article to be a further crucial piece in the medium-term agenda of realizing a truly distributed social network based on

[13] https://joindiaspora.com
[14] http://noserub.com
[15] http://www.gnu.org/software/social/
[16] An ordered list can obtained from Wikipedia: http://en.wikipedia.org/wiki/List_of_social_networking_websites

semantic technologies. Since mobile devices are playing an increasingly important role as clients and platforms for social networks, our realization focused on providing a extensible framework for social semantic networking on the Android platform. With this work we aimed at showcasing how different (social) Semantic Web standards, technologies and best practices can be integrated into a comprehensive architecture for social networking (on mobile devices).

With regard to future work we plan to further decrease the entrance barrier for ordinary users. A current obstacle is that users are required to have a WebID and - if they want to use authentication and access control features - a FOAF+SSL enabled WebID. In particular creating a FOAF+SSL enabled WebID is, due to the certificate creation, still a cumbersome process. A possible simplification of this process would be to enable mobile phone users to create and upload the required profile and certificates directly from their mobile device. We also plan to implement a more efficient and user-friendly way for subscribing to updates of contacts. These will include profile changes, status updates, (micro-)blog posts as well as updates retrieved from social networking apps. This feature would be facilitated by a proxy infrastructure, which caches updates until the device re-connects to the network after a period of absence (e.g. due to limited network connection or switched-off devices). A further important aspect to be developed is the standardization and realization of social networking applications, which seamlessly integrate with and run on top of the distributed social semantic network. Such applications would comprise everything we know from centralized social networks (e.g. games, travel, quizzes etc.), but would make use of FOAF+SSL and the other distributed social networking components for authentication, access control, subscription/notification etc.

Acknowledgments. This work was supported by a grant from the European Union's 7th Framework Programme provided for the project LOD2 (GA no. 257943).

References

1. 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)
2. Becker, C., Bizer, C.: Exploring the Geospatial Semantic Web with DBpedia Mobile. J. Web Sem. 7(4), 278–286 (2009)
3. Berners-Lee, T.: Long Live the Web. Scientific American (December 2010)
4. Brickley, D., Miller, L.: FOAF Vocabulary Specification. Namespace Document September 2, FOAF Project (2004), `http://xmlns.com/foaf/0.1/`
5. David, J., Euzenat, J.: Linked data from your pocket: The Android RDFContentProvider. 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, Springer, Heidelberg (2010)

6. Halpin, H., Tuffield, M.: A Standards-based, Open and Privacy-aware Social Web. W3C Incubator Group Report, W3C (December 2010)
7. Iannella, R., Halpin, H., Suda, B., Walsh, N.: Representing vCard Objects in RDF. W3c Member Submission, W3C (January 2010)
8. Langridge, S., Hickson, I.: Pingback 1.0. Technical report (2002), `http://hixie.ch/specs/pingback/pingback`
9. Lassila, O.: Using the Semantic Web in Mobile and Ubiquitous Computing. In: Proc. of the 1st IFIP WG12.5 Working Conference on Industrial Applications of Semantic Web, Jyväskylä, Finland. IFIP, vol. 188, pp. 19–25 (2005)
10. Lassila, O., Adler, M.: Semantic Gadgets: Ubiquitous Computing Meets the Semantic Web. In: Spinning the Semantic Web: Bringing the World Wide Web to Its Full Potential, pp. 363–376. MIT Press, Cambridge (2003)
11. Le-Phuoc, D., Parreira, J.X., Reynolds, V., Hauswirth, M.: RDF On the Go: An RDF Storage and Query Processor for Mobile Devices. In: Posters and Demos of the ISWC 2010, Shanghai, China (2010)
12. Passant, A., Mendes, P.: sparqlPuSH: Proactive notification of data updates in RDF stores using PubSubHubbub. In: SFSW 2010 (2010)
13. Schandl, B., Zander, S.: Adaptive RDF Graph Replication for Mobile Semantic Web Applications. Ubiquitous Computing and Communication Journal (Special Issue on Managing Data with Mobile Devices) (August 2009)
14. Schwagereit, F., Scherp, A., Staab, S.: Representing Distributed Groups with dgFOAF. In: Aroyo, L., Antoniou, G., Hyvönen, E., ten Teije, A., Stuckenschmidt, H., Cabral, L., Tudorache, T. (eds.) ESWC 2010. LNCS, vol. 6089, pp. 181–195. Springer, Heidelberg (2010)
15. Sonntag, D., Engel, R., Herzog, G., Pfalzgraf, A., Pfleger, N., Romanelli, M., Reithinger, N.: SmartWeb handheld — multimodal interaction with ontological knowledge bases and semantic web services. In: Huang, T.S., Nijholt, A., Pantic, M., Pentland, A. (eds.) ICMI/IJCAI Workshops 2007. LNCS (LNAI), vol. 4451, pp. 272–295. Springer, Heidelberg (2007)
16. Sporny, M., Corlosquet, S., Inkster, T., Story, H., Harbulot, B., Bachmann-Gmür, R.: WebID 1.0: Web identification and Discovery. Unofficial draft (August 2010), `http://payswarm.com/webid/`
17. Story, H., Harbulot, B., Jacobi, I., Jones, M.: FOAF+SSL: RESTful Authentication for the Social W. In: SPOT 2009 (2009)
18. Tramp, S., Frischmuth, P., Ermilov, T., Auer, S.: Weaving a Social Data Web with Semantic Pingback. In: Cimiano, P., Pinto, H.S. (eds.) EKAW 2010. LNCS (LNAI), vol. 6317, pp. 135–149. Springer, Heidelberg (2010)
19. Wilson, M., Russell, A., Smith, D.A., Owens, A., Schraefel, M.C.: mSpace Mobile: A Mobile Application for the Semantic Web. In: Gil, Y., Motta, E., Benjamins, V.R., Musen, M.A. (eds.) ISWC 2005. LNCS, vol. 3729, Springer, Heidelberg (2005)

Automatic Semantic Subject Indexing of Web Documents in Highly Inflected Languages

Reetta Sinkkilä, Osma Suominen, and Eero Hyvönen

Semantic Computing Research Group (SeCo),
Aalto University, Department of Media Technology
University of Helsinki, Department of Computer Science
firstname.lastname@tkk.fi
http://www.seco.tkk.fi/

Abstract. Structured semantic metadata about unstructured web documents can be created using automatic subject indexing methods, avoiding laborious manual indexing. A succesful automatic subject indexing tool for the web should work with texts in multiple languages and be independent of the domain of discourse of the documents and controlled vocabularies. However, analyzing text written in a highly inflected language requires word form normalization that goes beyond rule-based stemming algorithms. We have tested the state-of-the art automatic indexing tool Maui on Finnish texts using three stemming and lemmatization algorithms and tested it with documents and vocabularies of different domains. Both of the lemmatization algorithms we tested performed significantly better than a rule-based stemmer, and the subject indexing quality was found to be comparable to that of human indexers.

1 Introduction

The Semantic Web vision requires structured ontological metadata in order to provide novel services such as rich search interfaces, automatic recommendations, agent-based assistants and semantic personalization. The current Web, however, consists largely of unstructured text documents. Manually annotating such content is often infeasible due to the large amount of work involved, and in any case may not always produce good results [3].

One important method for creating structured descriptions of unstructured text is automatic *subject indexing*, also known as *term assignment*, which is the process of summarizing the content of a document by selecting multiple subjects from a controlled vocabulary that describe its topic [9,10]. Many automatic subject indexing tools exist for various languages and domains [18]. For example, many systems have been developed for automatically assigning subjects from the Medical Subject Headings (MeSH) vocabulary for biomedical documents [20].

A succesful automatic subject indexing tool for the Web should be flexible enough to work with documents in different languages and domains. The quality of the automatically assigned subjects can usefully be compared with traditional manual subject indexing. However, when two humans describe the same document, they are very unlikely to select the same subjects [24,8,16]. Thus,

© Springer-Verlag Berlin Heidelberg 2011

rather than relying on the subjects assigned by a single human indexer as a gold standard, it is more useful to compare the degree of consistency between an automated algorithm and several independent human indexers [10].

Subject indexing algorithms perform language analysis and normalization such as stemming [10,18]. However, in agglutinative and highly inflected languages such as Finnish [6], Turkish [12], Estonian, Hungarian and Slavic languages, a simple stemming strategy is unlikely to perform well [4,6].

In this paper, we set out to find a strategy for automatic subject indexing for inflectional languages, using Finnish as the test case with an intention to produce annotations of comparable quality to those produced by human indexers. Finding such a strategy would allow us to substantially increase the amount of structured metadata for use in Finnish Semantic Web portals such as HealthFinland[1] and CultureSampo[2]. In particular, we seek answers to the following research questions:

1. What kind of **stemming** or **lemmatization** strategy gives the best results when performing automatic subject indexing for web documents in highly inflected languages?
2. What is the **quality** of automatically assigned terms for documents written on inflected languages compared with human indexers?
3. Does the same automatic term assignment strategy work **independently of the domain** of the documents and vocabularies?

To answer these questions, we performed a series of automatic subject indexing experiments on Finnish language documents. We used the Maui indexing tool, a language- and domain-independent system which incorporates state-of-the-art topic ranking algorithms and can perform subject indexing using a controlled vocabulary [10]. To test the effect of morphological analysis strategies, we coupled the Maui tool with several available stemming and lemmatization tools. We compared the results of the automatic indexing with subjects assigned by human indexers. To compare the performance of the tools in different domains, we used document sets and vocabularies from two domains: a) documents from a social sector website together with a health-oriented ontology, and b) point of interest descriptions from Wikipedia together with a general purpose ontology.

Our results indicate that the Maui subject indexing algorihms work relatively well even with Finnish language documents when Maui is coupled with a capable lemmatization system, and the indexing quality is comparable to that of human indexers. However, disambiguation between similar or overlapping concepts in the vocabulary was problematic in some cases. Also, the handling of set phrases and compound words caused some issues. Some of the ambiguities might be resolved by using part-of-speech information as an aid in disambiguation. The results should generalize to other highly inflected and agglutinative languages. Even the subject indexing of English documents might be improved by performing more sophisticated linguistic analysis than simple rule-based stemming.

[1] http://www.tervesuomi.fi

[2] http://www.kulttuurisampo.fi

2 Related Work

Automatic subject indexing consists of two phases: performing linguistic analysis for matching document words or n-grams with meanings expressed as terms in a controlled vocabulary (*semantic tagging*) and determining which of the matched vocabulary terms best describe the document (*topic ranking*).

Semantic tagging. is the matching of words to meanings and a part of linguistic analysis. Linguistic analysis for the purpose of annotation consists of five steps: morphological analysis, part-of-speech tagging, chunking, dependency structure analysis and semantic tagging [1]. In languages such as English, Spanish and French, a simplified form of semantic tagging can be performed by using a rule-based stemming algorithm to normalize both document words and vocabulary terms [10]. This allows, e.g., singular words to be matched with plural terms in the vocabulary.

Inflected languages such as Finnish, Turkish, Arabic and Hungarian typically express meanings through morphological affixation. In highly inflected languages plural and possessive relations, grammatical cases, and verb tenses and aspects, which in English would be expressed with syntactic structures, are characteristically represented with case endings [12,6]. Compound words are also typical in inflected languages. Rule-based stemming does not work particularly well for semantic tagging: as an example, a semantic tagger for the Finnish language developed in the Benedict project used a sophisticated morphological analysis and lemmatisation tool as well as rules for handling compound words in order to attain high precision [6,7].

In **topic ranking**, machine learning methods have surpassed rule-based methods for determining the important topics of a document [18]. The TF$\times$IDF method provides a baseline [17], which Maui [10] and its predecessors KEA [23] and KEA++ [11] have improved on by additionally using various heuristics. These tools can also perform topic indexing without the support of a controlled vocabulary, known as *keyphrase extraction.* The previous Maui tests on English, French and Spanish docments have used a stemming algorithm for basic semantic tagging. In those languages, Maui has been found to assign subjects of comparable quality of those of human indexers [10].

KEA has been ported to support other languages. A Turkish adaptation of KEA was used to extract keyphrases and a controlled vocabulary was not used [13]. A KEA-like approach for keyphrase extraction of Arabic documents has also been found to perform well when part-of-speech analysis was incorporated into the candidate selection phase [2].

Other subject indexing tools for inflected languages include the Poka information extraction tool for Finnish [21], which has been used, e.g., in the Opas system to assign concepts from the Finnish General Upper Ontology to question-answer pairs [22]. The Leiki platform is a commercial tool that analyzes Finnish text and determines its important concepts using a proprietary ontology-like classification system [14]. It is used by many Finnish news websites for automatically generating links to related content. However, neither tool has been evaluated in academic literature.

3 Materials and Methods

For our experiments, we have used three document collections together with two vocabularies. With these, we performed three experiments using the Maui indexer and three different stemming and lemmatization tools.

3.1 Document Collections

To provide material for experiments we prepared two corpora and annotated them with different vocabularies. This was to ensure that we can measure the performance of the automatic indexing independently of the domain of the documents and vocabularies.

The first text corpus consists of documents extracted from the Sosiaaliportti web portal[3] maintained by the Finnish National Institute for Health and Welfare. Sosiaaliportti is designed for professionals in the social sector, and is intended to support social workers in their daily work. It contains 1) question-answer pairs on topics related to social work in general, such as "What are the criteria for granting a transportation service for a severly disabled person", 2) a discussion forum, 3) the handbook for child welfare which is intended to be used as a topical manual for professionals.

The first Sosiaaliportti document collection we used, **SOS-60**, consists of 60 randomly extracted documents from the Sosiaaliportti portal. This sample includes 30 documents from the Handbook for child welfare and 30 question-answer pairs from the consultancy service archive. The documents are relatively short, ranging from 33 to 1324 words with an average of 360 words.

The second document collection **SOS-30** is a subset of SOS-60, consisting of 15 question-answer pairs and 15 Handbook documents. It was created in order to determine the inter-indexer consistency of human indexers.

The document collections were indexed by employees of the National Institute for Health and Welfare – professionals ranging from a summer trainee to a medical doctor. Indexers were advised to use 3–8 subjects per document, which is the usual amount of index terms used in the National Institute for Health and Welfare content indexing process. The SOS-60 collection was indexed by a single person, who assigned an average of 5 subjects per document. The SOS-30 collection was indexed by six people, with an average of 4.1 subjects per document. The mimum number of assigned subjects was 0 (two indexers used this) and maximum number was 9. Summary of the number of subjects used by indexers is in table 1. Both datasets were created and indexed for the purpose of the experiments reported in this paper.

[3] `http://www.sosiaaliportti.fi`

Table 1. Number of subjects assigned to each document in SOS-30

	Min	Max	Mean	St. deviation
Indexer 1	0	6	2.8	1.6
Indexer 2	1	9	3.7	1.7
Indexer 3	0	8	4.6	1.7
Indexer 4	3	6	3.9	0.8
Indexer 5	2	8	4.4	1.6
Indexer 6	2	8	5.3	1.5
Average	1.3	7.5	4.1	1.5

To test the domain independence of the Maui topic extractor, another document collection **POI-61** was created, consisting of 61 documents extracted from the Finnish Wikipedia with subjects covering Finnish Points of Interest (POIs) such as churches and statues. Characteristically these documents are also relatively brief, containing 450 words per document on the average. The POI-61 collection was indexed by a single person. The average number of subjects per document was 7.6 with a minimum of 1 and maximum of 15 subjects.

The charasteristics of the document collections are slightly different. The Sosiaaliportti collections consist of shorter documents that are indexed with fever terms per document, while the Wikipedia corpus is more exhaustively annotated. Also the content and structure of the documents differs. Sosiaaliportti documents have been written by professionals and they cover topics more indepth. The Wikipedia documents are more of a descriptive nature.

3.2 Vocabularies

The Sosiaaliportti document collections were indexed using concepts from the Finnish Ontology of Health and Welfare, Tero. It is a combination of several health domain vocabularies including The European Multilingual Thesaurus on Health Promotion (HPMULTI)[4] and a subset the Medical Subject Headings (MeSH) thesaurus[5], merged with the Finnish General Upper Ontology YSO[6]. YSO is based on the Finnish General Thesaurus maintained by the National Library of Finland.

Tero defines over 24 000 concepts which have Finnish, Swedish and English labels. Only the Finnish labels have been used in the indexing process described in this research. There are also alternative labels for some concepts such that the total amount of Finnish labels in the ontology is around 30 000. Tero is represented in SKOS format and the relations between terms have been represented according to SKOS conventions, e.g. the skos:broader relation representing a hierarchical relation. Tero contains some ambiguous terms with disambiguating

[4] http://www.hpmulti.net/
[5] http://www.nlm.nih.gov/mesh/
[6] http://www.seco.tkk.fi/ontologies/yso/

context information coded in parenthesis, for example *lapset (perheenjäsenet)* and *lapset (ikäryhmä)* standing for *children (family members)*, and *children (age group)*, respectively. Some of these ambiguous concepts have the unqualified ambiguous labels as alternative label, e.g. *lapset*.

The POI-61 documents were indexed using the Finnish Collaborative Holistic Ontology, KOKO. It is a collection of Finnish core ontologies that have been merged together. The ontologies include the Finnish General Upper Ontology YSO as its top ontology and a variety of other domain specific ontologies extending its concepts into more detailed subconcept hierarchies. These include for example the ontology for museum domain, the ontology for applied arts and the Finnish ontology for photography. KOKO defines some 30 000 concepts and is encoded in SKOS format. Concepts have preferred and alternative labels in Finnish, Swedish and English. Only the Finnish labels were utilized in these experiments.

Table 2. Sizes of the TERO and KOKO vocabularies

	Total terms	PrefLabels	AltLabels	Ambiguous
Finnish TERO	30,040	24,270	5,770	1720
Finnish KOKO	38,690	30,080	7,810	1910
English Agrovoc	38,200	28,170	10,030	400
French Agrovoc	37,350	28,160	9,190	440
Spanish Agrovoc	40,640	28,160	12,480	620

A summary of vocabularies used in this research is shown in table 2. For comparison, the table also includes the corresponding statistics of the Agrovoc thesaurus which was used in the original Maui experiments [10]. Both TERO and KOKO contain a relatively large number of ambiguous terms compared to Agrovoc. This is due to the inclusion of the Finnish Upper General Ontology, which contains a large amount of everyday terms which more often have several meanings than domain-specific specialist terminology such as Agrovoc terms.

3.3 Maui Topic Indexing Tool

We selected the Maui topic indexing tool, version 1.2, for our automatic indexing experiments as it implements a state-of-the art topic ranking algorithm [10]. Although Maui can be used without a controlled vocabulary, we will concentrate on the case when a vocabulary is used. The topic ranking is based on a number of heuristics (called *features* in Maui terminology) including TF×IDF, spread (separation of first and last occurrence), semantic relatedness based on vocabulary structure, and term length. The algorithm is first tuned with a small training set of manually indexed documents, which is used to tune the relative weights of the heuristics. After training has been completed, it can perform subject indexing on new documents.

In the indexing phase, Maui first splits the text into textual segments (usually sentences). These are then further split into words, which are then grouped into n-grams. The n-grams are matched with terms in a controlled vocabulary; stemming is performed both on the n-grams and the vocabulary terms in order to increase recall, for example by matching singular form words with plural forms in the vocabulary. The stemming algorithms are language specific, but new stemmers can be plugged in.

Finally, the n-grams which were determined to match vocabulary terms are ranked by applying the different heuristics and summarizing the feature values according to the weights that were determined in the training phase. The top K matched vocabulary terms are assigned as subjects for the document.

3.4 Stemming and Lemmatization Tools

We inspected the effect of different word form normalizations by testing three methods for deriving base forms of words. We tested the commercial syntactic dependency parser FDG version 3.8.1 for Finnish [19] by Connexor Ltd, the morphological analyzer Omorfi version 20100401 for Finnish [5] and the Snowball stemmer for Finnish[7].

The main difference between these tools is that while FDG and Omorfi try to reduce the word forms into their lemmas base forms, the Snowball stemmer only stems word by cutting off inflectional suffixes without fixing the consonant gradiation. Another difference between the selected tools is how they handle compound words and set phrases as well as words unfamiliar to them. For example, the word *seurakuntatyö* (*church/parish work*) is a coumpound word consisting of parts *seurakunta* and *työ*, with meanings *parish* and *work*, respectively. The word *seurakunta* could also be split into its constituent parts *seura* and *kunta*, but the compound word has a special lexicalized meaning which does not directly follow from the parts. The FDG parser lemmatizes the word correctly recognizing the fixed compound semantic meaning and handles the word as a compound word. The version of the Omorfi parser we used instead returns every possible combination of the word parts without any weights indicating some interpretation as more favorable. With unfamiliar words Omorfi returns the original input, whereas FDG and Snowball always try to reduce the word to a base form.

3.5 Experiments

We conducted three sets of experiments in order to answer our research questions. The purpose of the first experiment was to determine the effect of lemmatization method used. The second experiment compares the quality of automatically assigned terms with human indexing. The last test set tries to evaluate whether the automatic term assignment strategy works independently of the domain of the documents and vocabularies.

[7] http://snowball.tartarus.org/

Stemming and Lemmatization Strategy. The first experiment was to test how well Maui performs with Finnish data using different stemming and lemmatization strategies. We used the SOS-60 document collection and the Tero vocabulary for this experiment. To provide a useful point of comparison, the experiment setup closely followed the experiment described in [10], section 7.2.4. *Language independence*, a test conducted with collections of 67 French and 47 Spanish agricultural documents indexed with Agrovoc thesaurus terms.

The test settings were the following: the stemmer was set to either FDG, Omorfi or Snowball. The stopword list was set to a list of Finnish stopwords taken from the Snowball string processing language site[8]. Document encoding was set to UTF-8. Document language was set to *fi*, to use the Finnish labels of the vocabulary. Tests were conducted with the leave one out technique. That is, the maximum possible number of documents (59) were used to create a model and the one remaining document was used for testing the model. The tests were repeated 60 times, each time indexing a different document. This is the same approach which was taken in the original tests with Spanish and French documents [10]. For each document 5 terms were extracted, which was the average number of manually assigned subjects per document in SOS-60. We re-ran the tests using each stemming or lemmatization tool in turn.

In addition, to test the effect of the Maui topic ranking algorithms, we used the term frequency – inverse document frequency method TF×IDF as a baseline by turning off all other Maui topic ranking algorithms. It was calculated only using terms from the vocabulary as candidates, and FDG as a lemmatizer.

Inter-Indexer Consistency. In the second experiment, we tested how well Maui performs related to human indexers. We used the SOS-30 document collection, indexed by six independent people, and the Tero vocabulary. To provide a reference for evaluation, we first measured the performance of the independent human annotators by measuring the similarity between indexers. We measured the consistency with the Rolling measure [15], defined as

$$\frac{2C}{A+B} \tag{1}$$

where C is the amount of subjects two indexers have in common and A and B the amount of subjects used by indexer A and B respectively. With this measure, two identically annotated documents get a similarity value of 1 and totally distinct annotations get a value of 0. We counted this measure for each document between every indexer-indexer pair. The total consistency between two indexers is the average of the document specific values.

We then indexed the same document set with the Maui topic extraction tool using each human annotated document set as training material in turns. Similarly to the first experiment, this was done with the leave-one-out technique to maximize the available training material. For each document 4 subjects were

[8] http://snowball.tartarus.org/algorithms/finnish/stop.txt

assigned, which was the average number of terms the human indexers had assigned per document in SOS-30. We used FDG to perform lemmatization in this experiment. The other parameters were the same as in the first experiment.

Automatic annotations were then compared to the manually made annotations, first in pairs with each annotator and finally the average agreement was calculated with the same procedure as that used between human indexers. This made it possible to directly compare the performance of Maui to a human indexer.

Domain Independence. To test the domain independence and suitability of the method for different materials, we conducted a third experiment with a different document collection and vocabulary. The POI-61 document collection and Koko vocabulary were used in this experiment. The test was conducted in a similar way to the first experiment, but using only FDG for lemmatization.

4 Results

In this section, we present the results of the three experiments described above.

4.1 Stemming and Lemmatization Strategy

The first test setting was to test the suitability of Maui tool for Finnish language with alternating stemmers. The Maui topic extractor was ran with vocabulary language set to Finnish and stemmers set to FDG, Omorfi and Snowball in turns. We used the SOS-60 collection which is indexed with the Tero ontology.

Table 3. Stemming and lemmatization strategy results

	Precision	Recall	F-Measure
SOS-60, FDG	40.0	37.1	38.5
SOS-60, Omorfi	40.0	35.9	37.8
SOS-60, Snowball	35.7	32.2	33.8
SOS-60, FDG, TF×IDF only	27.0	24.4	26.7
French Agrovoc	34.5	31.8	33.1
Spanish Agrovoc	24.7	26.9	25.7

Compared to the baseline method, TF×IDF, the Maui topic extraction algorithm performed better regardless of which parser was used (Table 3). For comparison, the figures from the Maui tests with French and Spanish documents [10] are also included in the table. The best lemmatisation strategy was FDG but Omorfi also showed good results. The precision was the same for both strategies, but the ones performed with FDG resulted in better recall.

Two documents indexed with different methods are shown in Table 4. For each document terms assigned by a professional indexer and by Maui tool with FDG, Snowball and Omorfi parser, respectively, are presented. The first document is an example of a succesful automatic annotation. Annotations made with Maui tool with all of the lemmatisation strategies performed well compared to the human indexer. Correct terms are emphasized, inapplicable or redundant terms are marked with cursive text.

The second document is an example of an unsuccesful annotation procedure with some peculiarities. The human indexer has assigned topics related to Romani people, clothes and pregnancy. Automatically assigned terms include term *wood chip*, which is inapplicable with regards to the document's contents. This is a result of imperfect morphological analysis. That is, the document's contents have been processed with a stemmer, which has produced a stem, which has in turn been connected to a different term with the same stem. Also, some identical topics have been chosen to describe the same document. Maui tool has assigned terms *nuoret* (*young people*) and *nuori (13-18)* (*adolescent*), with overlapping meaning, the only difference being the first one in plural form and the second one accompanied by the specification *13-18*.

Table 4. Example documents

	Document 1 (good performance)		Document 2 (bad performance)	
	Finnish	English	Finnish	English
Human Indexer	perhehoito	foster care	vaatteet	clothes
	kiireellinen sijoitus	high-priority placement	raskaus	pregnancy
	huostaanotto	placement into care	romanit	Romani people
	avohuollon tukitoimet	support in community care	romanit - kulttuuri	Romani culture
	jälkihuolto	after-care		
	laitoshoito	institutional care		
	sijaishuolto	substitute care		
Maui + FDG	**sijaishuolto**	**substitute care**	*nuori (13-18)*	*adolescent*
	avohuollon tukitoimet	**support in community care**	ohjeet	instructions
	jälkihuolto	**after-care**	hakemukset	applications
	lapset (sosioek.)	children (socioeconomic)	**raskaus**	**pregnancy**
	huostaanotto	**placement into care**	*nuoret*	*young people*
Maui + Omorfi	**sijaishuolto**	**substitute care**	synnytys	delivery
	lapset (sosioek.)	children (socioeconomic)	**raskaus**	**pregnancy**
	jälkihuolto	**after-care**	nuori (13-18)	adolescent
	huostaanotto	**placement into care**	ohjeet	instructions
	avohuollon tukitoimet	**support in community care**	*kaulukset*	*collar*
Maui + Snowball	**sijaishuolto**	**substitute care**	nuori (13-18)	adolescent
	lapset (sosioek.)	*children (socioeconomic)*	hakemukset	applications
	jälkihuolto	**after-care**	naiset	women
	avohuollon tukitoimet	**support in community care**	*hake*	*wood chip*
	lapsi (6-12)	*child (6-12)*	*kunnat*	*municipalities*

4.2 Inter-indexer Consistency

The results of the inter-indexer consistency experiment are shown in Table 5 using the Rolling measure. The consistency between any two indexers can be found in the table. The average consistency of the human indexers was 33.7%. This corresponds to 22.6% on Hooper's scale [24]. Previous studies have reported inter-indexer consistencies to vary widely subject to, e.g., the previous experience of the indexers and the usage of controlled vocabulary [8]. [16] reports consistency close to 20%, while [8] presents consistencies between 10–80%.

There is some variance between the indexers. Indexer 1 is least consistent with the other human indexers (27.4%) while Indexer 5 agrees the most with the other indexers (36.6%).

Table 5. Consistency of human indexers 1–6 compared to Maui

	1	2	3	4	5	6	Average	**Maui**
1		25	29	28	27	28	27.4	**21.5**
2	25		31	30	36	37	31.8	**29.9**
3	29	31		40	42	39	36.2	**27.2**
4	28	30	40		38	35	34.2	**36.3**
5	27	36	42	38		40	36.6	**25.3**
6	28	37	39	35	40		35.8	**27.2**
							33.7	**27.9**

The Maui topic indexing algorithm is 27.9% consistent with human indexers. Maui indexes most alike with the Indexer 4 (36.3%) and least alike the with Indexer 1 (21.5%). There is quite a lot of variance between the performance of the automatic annotation method when compared with different indexers. The Maui indexer acts poorly with Indexer 1's document collection as training material. This might result from Indexer 1 using fewer than three terms per document, the average being four terms.

4.3 Domain Independence

The results of the third experiment using POI-61 and Koko were similar to those of the first SOS-60 collection test (Table 6), with slightly higher precision and recall values attained.

Table 6. Domain independence experiment results

	Precision	Recall	F-Measure
SOS-60 with FDG	40.0	37.1	38.5
POI-61 with FDG	45.4	38.1	41.4

5 Discussion and Future Work

In this section, we revisit the research questions based on the results, highlight some problems we encountered and present opportunities for future work.

What Kind of Stemming or Lemmatization Strategy Gives the Best Results When Performing Automatic Subject Indexing for Web Documents in Highly Inflected Languages? Our first experiment with three different stemming and lemmatization methods demonstrated that both Omorfi and FDG can be used for lemmatization and both will give good results. The simple rule-based Snowball stemming algorithm did not work as well.

The quality of indexing using Omorfi was almost as good as with FDG. This may at first be surprising, because Omorfi only analyzes the morphological structure of individual words, which may be ambiguous. In contrast, FDG is able to perform part-of-speech and dependency structure analysis. However, in this case the way Maui chunks sentences into individual words before stemming prevents FDG from seeing the whole sentence, thus making it impossible for FDG to analyze the word context. This presents an opportunity for future work: if Maui were adapted so that it is able to pass full sentences to the stemming algorithm, better indexing quality might be attained when using a more sophisticated lemmatization algorithm such as FDG. This kind of experiments could also be performed with less inflected languages such as English.

What is the Quality of Automatically Assigned Terms for Documents Written on Inflected Languages Compared with Human Indexers? The inter indexer consistency test found consistency between indexers to be 33.7%, whereas consistency between Maui and the human indexers was 27.9%. Maui annotates topics almost as well as human indexers, but there are rather large differences between indexers. The performance of Maui in terms of agreement with human indexers was slightly higher than that of Indexer 1, who had the lowest agreement score (27.4%). The result is somewhat better than in a previous similar evaluation, where the performance of Maui was lower than that of every professional human indexer [10, Table 7.7].

Human indexers are notoriously unreliable when unmotivated, for example taking shortcuts when asked to perform topical indexing as part of a publication process [3]. In our second experiment, some indexers used much fewer subjects per document than they were asked for, and left some documents unindexed. An automated algorithm may not perform as well as motivated professional indexers, but its results can be expected to be more consistent with the task specification.

Does the Same Automatic Term Assignment Strategy Work Independently of the Domain of the Documents and Vocabularies? It has previously been shown that the Maui algorithm works with documents and vocabularies of different domains, including the medical, physics and agriculture domains [10]. The results of our third experiment using point of interest descriptions and a general ontology suggest that when a suitable lemmatizer is used the algorithm also works well with Finnish text of different domains.

Problems Encountered. The most essential problem we encountered with the topic indexing with the selected methods, were related to disambiguation of the vocabulary terms. The Maui tool sometimes selected overlapping topics (Table 4) and was not always able to disambiguate between different meanings even though a semantically linked vocabulary was available. Especially problems arise when concepts share equal labels, which often happens with alternative labels in general purpose ontologies. This issue was especially pronounced when stemming was used instead of more sophisticated morphological analysis.

Further problems arose with set phrases, which the Maui tool can not handle as a unit. If the document contains terms *tunnustelu* (examination) and *käsi* (hand), Maui may assign it to a compound term from the vocabulary *käsin tunnustelu* (examination with hands).

Some disambigation problems might be avoided if the words of the documents were not considered in a bag of words style, where the possibility to disambiguate words based on part of speech or dependency structure is lost. If documents were sent to a syntactic dependency parser sentence by sentence, then some misinterpretations could be avoided.

Future Work. There is still room for improvement in topic indexing for inflective languages, particularly by using sentence-level analysis and part-of-speech disambiguation as discussed above. We are also looking at ways to simplify the use of information extraction methods for the automatic annotation of text documents that can then be incorporated into Semantic Web portals. We have produced an initial prototype of ARPA, which is an automatic annotation system that provides API access similar to the OpenCalais toolkit[9]. When completed, ARPA will feature a subject indexing facility based on Maui as well as an ontology-based named entity recognition facility.

6 Conclusion

A good automatic subject indexing algorithm makes it possible to substantially increase the amount of structured metadata on the Semantic Web. However, most research to date has concentrated on English language documents, where the language analysis can be performed by a simple rule-based stemmer.

Automatic subject indexing using stemmers is difficult in inflective languages such as Finnish, Turkish, Arabic and Slavic languages. In our experiments on Finnish documents and vocabularies using the Maui indexing toolkit, we were able to increase indexing quality by using more sophisticated lemmatization

[9] http://www.opencalais.com/

algorithms Omorfi and FDG instead of a simple rule-based stemmer. Using similar analysis tools would be useful for subject indexing other heavily inflected languages.

The subject indexing quality we attained was comparable to that of human indexers, in line with earlier similar experiments on documents in other languages. Indexing quality might yet be improved by using part-of-speech information and dependency structure analysis in the semantic tagging phase. Also, such a strategy might assist in disambiguating between similar concepts in a controlled vocabulary. However, even without these enhancements, the current quality of automated subject indexing is sufficient for performing many tasks that previously have relied on laborious human annotation.

Acknowledgements

This work is a part of the national semantic web ontology project FinnONTO[10] 2003-2012, funded mainly by the National Funding Agency for Technology Innovation (Tekes). We thank Hanna Heikkonen and Tuomas Palonen for organizing and coordinating the human indexing of test document sets.

References

1. Buitelaar, P., Declerck, T.: Linguistic Annotation for the Semantic Web. In: Annotation for the Semantic Web, pp. 93–110. IOS Press, Amsterdam (2003)
2. El-Shishtawy, T., Al-Sammak, A.: Arabic Keyphrase Extraction using Linguistic knowledge and Machine Learning Techniques. In: Proceedings of the Second International Conference on Arabic Language Resources and Tools (2009)
3. Hawking, D., Zobel, J.: Does Topic Metadata Help With Web Search? Journal of the American Society for Information Science and Technology 58(5), 613–628 (2007)
4. Hirsimäki, T., Creutz, M., Siivola, V., Kurimo, M., Virpioja, S., Pylkkönen, J.: Unlimited Vocabulary Speech Recognition with Morph Language Models Applied to Finnish. Computer Speech & Language 20(4), 515–541 (2006)
5. Lindén, K., Silfverberg, M., Pirinen, T.: HFST Tools for Morphology – An EfficientOpen-Source Package for Construction of Morphological Analyzers. In: Mahlow, C., Piotrowski, M. (eds.) State of the Art in Computational Morphology. CCIS, vol. 41, pp. 28–47. Springer, Heidelberg (2009)
6. Löfberg, L., Archer, D., Piao, S., Rayson, P., Mcenery, T., Varantola, K., pekka Juntunen, J.: Porting an English semantic tagger to the Finnish language. In: Proceedings of the Corpus Linguistics 2003 Conference (2003)
7. Löfberg, L., Piao, S., Nykanen, A., Varantola, K., Rayson, P., Juntunen, J.P.: A semantic tagger for the Finnish language. In: Proceedings of Corpus Linguistics 2005 (2005)
8. Markey, K.: Interindexer Consistency Tests: A Literature Review and Report of a Test of Consistency in Indexing Visual Materials. Library and Information Science Research, An International Journal 6(2), 155–177 (1984)

[10] http://www.seco.tkk.fi/projects/finnonto/

9. Maron, M.E.: Automatic Indexing: an Experimental Inquiry. Journal of the ACM (JACM) 8(3), 404–417 (1961)
10. Medelyan, O.: Human-competitive automatic topic indexing. Ph.D. thesis, University of Waikato, Department of Computer Science (2009)
11. Medelyan, O., Witten, I.H.: Thesaurus Based Automatic Keyphrase Indexing. In: Proceedings of the 6th ACM/IEEE-CS Joint Conference on Digital Libraries (2006)
12. Oflazer, K., Kuruöz, I.: Tagging and Morphological Disambiguation of Turkish Text. In: Proceedings of the Fourth Conference on Applied Natural Language Processing (1994)
13. Pala, N., Çiçekli, I.: Turkish Keyphrase Extraction Using KEA. In: Proceedings of the 22nd International Symposium on Computer and Information Sciences, ISCIS 2007 (2007)
14. Pennanen, P., Alatalo, T.: Leiki – a platform for personalized content targeting. In: Proceedings of the 12th ACM Conference on Hypertext and Hypermedia, HYPERTEXT 2001 (2001)
15. Rolling, L.: Indexing consistency, quality and efficiency. Information Processing & Management 17(2), 69–76 (1981)
16. Saarti, J.: Consistency of subject indexing of novels by public library professionals and patrons. Journal of Documentation 58(1), 49–65 (2002)
17. omorfi Salton, G., Buckley, C.: Term-weighting Approaches in Automatic Text Retrieval. Information Processing and Management 24(5), 513–523 (1988)
18. Sebastiani, F.: Machine Learning in Automated Text Categorization. ACM Computing Surveys 34(1), 1–47 (2002)
19. Tapanainen, P., Järvinen, T.: A non-projective dependency parser. In: Proceedings of the Fifth Conference on Applied Natural Language Processing (1997)
20. Trieschnigg, D., Pezik, P., Lee, V., de Jong, F., Kraaij, W., Rebholz-Schuhmann, D.: MeSH Up: Effective MeSH Text Classification for Improved Document Retrieval. Bioinformatics 25(11), 1412–1418 (2009)
21. Valkeapää, O., Alm, O., Hyvönen, E.: Efficient content creation on the semantic web using metadata schemas with domain ontology services (System description). In: Franconi, E., Kifer, M., May, W. (eds.) ESWC 2007. LNCS, vol. 4519, pp. 819–828. Springer, Heidelberg (2007)
22. Vehviläinen, A., Hyvönen, E., Alm, O.: A semi-automatic semantic annotation and authoring tool for a library help desk service. In: Emerging Technologies for Semantic Work Environments: Techniques, Methods, and Applications, pp. 100–114. IGI Group, Hershey (2008)
23. Witten, I.H., Paynter, G., Frank, E., Gutwin, C., Nevill-Manning, C.G.: KEA: Practical Automatic Keyphrase Extraction. In: Proceedings of Digital Libraries 1999 (1999)
24. Zunde, P., Dexter, M.E.: Indexing Consistency and Quality. American Documentation 20(3), 259–267 (1969)

FootbOWL: Using a Generic Ontology of Football Competition for Planning Match Summaries

Nadjet Bouayad-Agha[1], Gerard Casamayor[1], Leo Wanner[1,2], Fernando Díez[3], and Sergio López Hernández[3]

[1] DTIC, University Pompeu Fabra, Barcelona, Spain
firstname.lastname@upf.edu
[2] Catalan Institute for Research and Advanced Studies (ICREA)
firstname.lastname@icrea.es
[3] DII, Universidad Autónoma de Madrid, Madrid, Spain
firstname.lastname@uam.es

Abstract. We present a two-layer OWL ontology-based Knowledge Base (KB) that allows for flexible content selection and discourse structuring in Natural Language text Generation (NLG) and discuss its use for these two tasks. The first layer of the ontology contains an application-independent base ontology. It models the domain and was not designed with NLG in mind. The second layer, which is added on top of the base ontology, models entities and events that can be inferred from the base ontology, including inferable logico-semantic relations between individuals. The nodes in the KB are weighted according to learnt models of content selection, such that a subset of them can be extracted. The extraction is done using templates that also consider semantic relations between the nodes and a simple user profile. The discourse structuring submodule maps the semantic relations to discourse relations and forms discourse units to then arrange them into a coherent discourse graph. The approach is illustrated and evaluated on a KB that models the First Spanish Football League.

1 Introduction

Natural language generators typically use as input external or purpose-built domain databases (DBs) or knowledge bases (KBs), extracting and/or transforming the relevant content during the text planning phase to instantiate schemas or other discourse representations, which are then verbalized during linguistic generation. See, for instance, [9]. More recent statistical, or heuristic-based, text planning tends to draw upon KBs crafted specifically for the task of Natural Language Generation (NLG) in order to assess relevance of its parts for inclusion into the text plan; see, among others, [4,5]. Given the NLG-tuned nature of these KBs, the mapping from knowledge to linguistic representations is then quite straightforward.

© Springer-Verlag Berlin Heidelberg 2011

In order to avoid linguistically-driven projection of relevant content onto discourse representations that intermingles conceptual information with linguistic information or the creation of NLG-tuned KBs, we suggest a two-layer KB. The first layer consists of a base ontology modeled in OWL. This ontology is application-independent: it only models the domain and was not designed with NLG in mind. The second layer, which is added on top of the base ontology, models entities and events that can be inferred from the base KB, including logico-semantic relations that can be inferred between individuals. Evidence on the existence of the inferred individuals and relations between them is deduced from a reference text collection. The KB used in our experiments models Football Competitions, more specifically the First Spanish Football League.

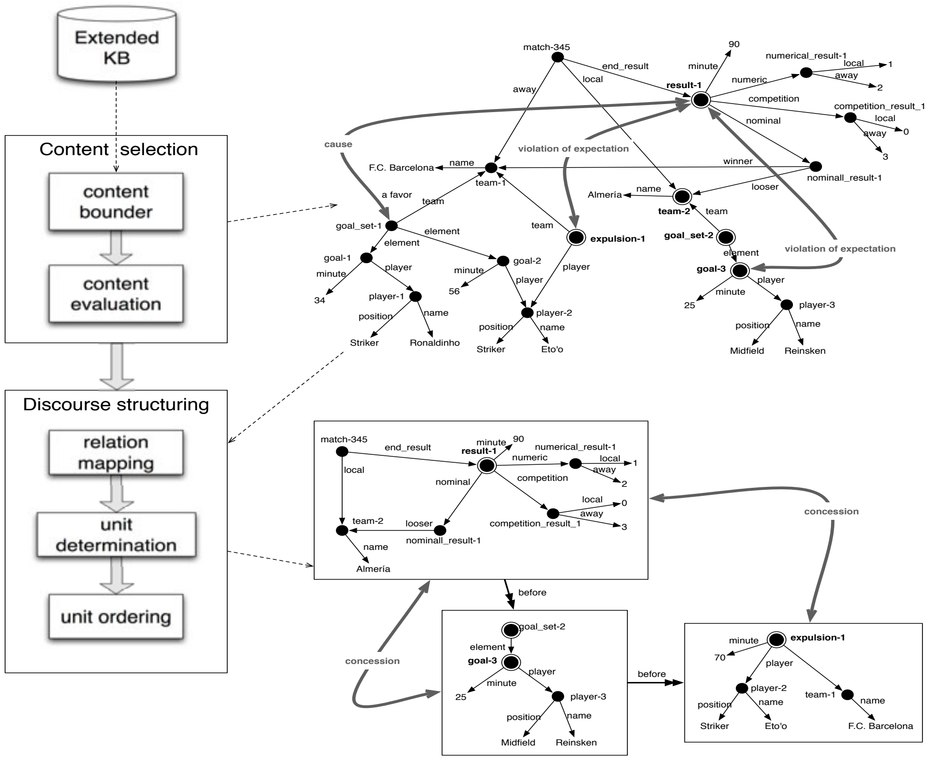


Fig. 1. Overview of text planning with associated content (top) and discourse (bottom) structures

In what follows, we describe how this KB is used for the two tasks of text planning, content selection and discourse structuring (Figure 1 illustrates the overall picture). Given their interconnection, these tasks are performed in an interplay between the content selection and discourse structuring modules. Thus, the relevance of the nodes in the KB is determined in the content selection module according to a simple user model, a set of relevance heuristics based on empirically determined weights, and the availability of logico-semantic relations that link the nodes to form a coherent content structure. The discourse representation module maps the logico-semantic relations to discourse relations, extracts the content marked as relevant during content selection using a set of templates to instantiate discourse units and to arrange them into a coherent discourse structure. The discourse structure is passed to the linguistic generator of Spanish, which produces short summaries of football matches of the kind found at the beginning of exhaustive articles about individual matches, only that they take the preferences of the targeted addressee for one of the teams involved into account. Consider Figure 2 that displays a generated summary that targets a fan of the team of Barcelona.[1]

Victoria del F.C. Barcelona. El Barcelona ganó contra el Almería por 2-1 gracias a un gol de Ronaldinho en el minuto 34 y otro de Eto'o en el minuto 56. El Barcelona ganó aunque acabó el partido con 10 jugadores a causa de la expulsión de Eto'o. Gracias a esta victoria, permanece en la zona de champions. En la vigésimo quinta jornada, se enfrentará al Villarreal.

Fig. 2. A sample football match summary as produced by our generator

The remainder of the paper is structured as follows. In Section 2, we present the two-layer ontology used as input to the text planning. In Section 3, we detail our content selection module, and present how we empirically determine weights using supervised learning to assess the relevance of some of the content units found in football match summaries. Section 4 describes the discourse structuring module and Section 5 the evaluation of the content selection and discourse structuring modules. Section 6 presents related work on the use of knowledge bases, data assessment, text planning and empirical content selection in NLG, before in Section 7 some conclusions are given and plans for further work are outlined.

2 A Two-Layer OWL Ontology

2.1 Ontology Design

There are some ontologies available that deal with sports and, more precisely, with (European) football (or soccer).[2] However, even the most detailed of them,

[1] Translation: 'Victory of F.C. Barcelona. Barcelona won against Almería by 2-1 thanks to a goal by Ronaldinho in minute 34 and another goal by Eto'o in minute 56. Barcelona won despite ending the match with 10 players because of the sent off of Eto'o. Thanks to this victory, Barcelona remains in the Champions zone (of the classification). Gameweek 25 Barcelona will meet Villareal.'

[2] Among them `http://sw.deri.org/ knud/swan/ontologies/soccer` (the SWAN Soccer Ontology by DERI), the sports fragment of the OpenCyc Ontology [12] (`http://sw.opencyc.org/2009/04/07/concept/en/Soccer`), the sports fragments in the DAML repository [7] (`http://www.daml.org/ontologies/374`) and [18].

let alone generic ontologies such as OpenCyc, did not contain specific football data we were interested in—among them, Approximation, Shot, Block or Header. Therefore, we developed our ontologies from scratch.

As already mentioned in the Introduction, our model foresees a two-layer ontology, the base ontology and the extended ontology.[3] The base ontology describes the football league domain. It is composed of two different ontologies: an object ontology which deals with the structural information of the competition (teams, competition phases, matches, players, etc.), and an event ontology which deals with information related to the events that happen in the match (penalties, goals, cards, etc.). To develop it, we followed the top-down strategy suggested by Uschold and King [19]: the more abstract concepts are identified first and subclassified then into more specific concepts. This is done to control the level of detail wanted. A known drawback of this strategy is that it can lead to an artificial excess of high-level classes. In our application, we achieved a sufficient level of detail for our application domain (i.e., the First Spanish Football League) with a moderate number of classes. More precisely, we model in the object ontology 24 classes and 42 properties, with 4041 instances in the corresponding KB (see Subsection 2.2 below). The top level classes of this ontology are Competition, Match, Period, Person, Result, Season, Team, TeamCompositionRelation and Title. In the event ontology, we model 23 classes and 8 properties, with 63623 instances in the corresponding KB (see Subsection 2.2 below). The top level classes of this ontology are ActionFault, Card, Corner, Fault, FaultKick, Goal, GoalKick, Interception, OffSide, Pass, Stop, Throw-in, Shot and Substitution.

The extended ontology adds an extra layer of meaning to the concepts modeled in the base ontology. Its concepts are deduced by the analysis of the target summaries, considering mainly what new knowledge can be inferred from the basic knowledge on the First Spanish Football League. We infer new knowledge about events and states of a match (goals and expulsions, results and classifications) typically found in summaries, excluding statistical information about matches within a season and across seasons (best scorer, consecutive wins, first victory in a given stadium, etc.). Some of the classes and properties were also added to make the navigation easier for the mapping to linguistic realization and for the inference of new knowledge. For example, 'for' and 'against' properties were added to the Goal class in order to know the team which scored respectively received the goal in case the information concerning team scored the goal was only available indirectly in the base ontology via the player. The inferred knowledge is divided into five categories, 1. result, 2. classification, 3. set, 4. match time, and 5. send-offs.

Result-related knowledge (nominal result and the points scored in the competition) is inferred from the numerical result of the match available in the base ontology (with winner/loser/drawing opponents specified), hence the classes NominalResult and CompetitionResult.

[3] The ontologies and corresponding knowledge bases are not available for free distribution. They are restricted to the i3media project consortium `https://i3media.barcelonamedia.org/`.

Classification-related knowledge models information related to the position of each team in the competition, its accumulated points and relative zone. For the zone, in addition to the four official zones Champions, UEFA, neutral or relegation, we introduce two internal zones—Lead and BottomOfLeague. It is of interest to obtain after each gameweek a team's tendency (ascending, descending, stable) and distance with respect to its previous classification. Tendency represents the team's change of zone in the competition, whilst Distance represents a team getting closer (or further) to a higher (lower) zone. In addition to the real tendency, teams are assigned a virtual tendency which represents the team's change of zone taking a (virtual) result that may be different from the actual match result (for instance, if the team would have drawn instead of winning, what would be the tendency of its classification in the league table).

Set-related knowledge models sets of events or processes for a given team in a match or for a given match. It is needed to be able to talk about events or processes together in accordance with their chronological occurrence (first goal, team was winning then it drew, etc.), hence the classes Set and ConstituentSet. These classes also allow us to simply refer to the number of constituents within it (cf. *the team had two red cards*).

Match time-related knowledge models the state of the match along its duration, creating intermediate results after each goal, hence the class IntermediateResult. Thus, a team could be winning after a goal, even though the final result is a draw. It is also possible to refer to specific reference time points such as 'beginning of the match', and 'conclusion of the first period'.

Send-offs related knowledge includes the expulsion of a player after a red card, hence the Expulsion class and the number of players left after an expulsion, hence the PlayersInField class.

Each set of inferred knowledge triggers the inference of a number of logico-semantic relations, hence the class LogicoSemanticRelation with its subclasses such as Cause, Implication, ViolationOfExpectation, Meronymy, Precedence, Contrast. For instance:

- A cause relation is instantiated between the set of goals of a team and the final nominal result.
- A violation-of-expectation relation is instantiated between an instance of PlayersInField and a final winning/drawing result (e.g., *despite playing with 10, the team won*).
- A relation of precedence is instantiated between pairs of constituents in a set to show their immediate temporal precedence relation.
- A contrast relation is instantiated between the contrasting classification distances or tendencies of both teams of the match (e.g., *team A goes up in the classification whilst team B goes down*).

Figure 3 shows the representation of the set of four goals of a team in a match, including the precedence relation between the constituents; the figure also shows the division of concepts between the base and extended ontology.

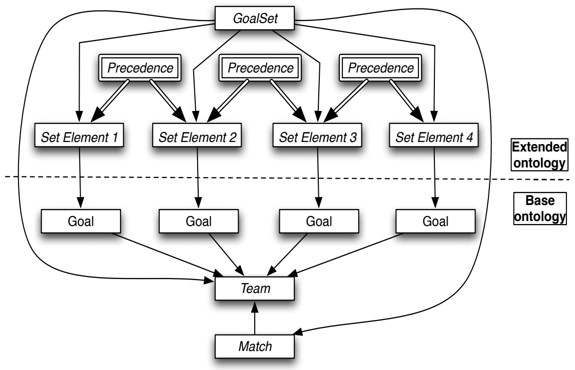


Fig. 3. Representation of an ordered set of goals of a team in a match

2.2 Ontology Population

The base KB was automatically populated with data scraped from web pages about the Spanish League seasons to include general information about competitions, players, stadiums, etc, and specific information about matches. Currently, it contains three seasons: 2007/2008, 2008/2009 and 2009/2010. The scrapping was done by *ad hoc* programs that extract all the information required by the classes defined in the base ontologies.[4] The extended ontology population was carried out using the inference engine provided by Jena.[5] The engine works with a set of user-defined rules consisting of two parts: head (the set of clauses that must be accomplished to fire the rule) and body (the set of clauses that is added to the ontology when the rule is fired). We defined 93 rules, with an estimated average of 9,62 clauses per rule in the head part. Consider the following example of a rule for classifying the difference between the scores of the two teams as "important" if it is greater than or equal to three:

```
[rule2: (?rn rdf:type base:NumericResult)
(?rn base:localScore ?localScore) (?rn base:visitorScore ?visitorScore)
(?localScore base:result ?local) (?visitorScore base:result ?visitor)
differenceAbs(?local, ?visitor, ?r) ge(?r, 3)
-> (?rn inference:resultDifference "important")]
```

For the 38 gameweeks of the regular football season, the inference engine generates using the 93 rules from the data in the base ontologies a total of 55894 new instances. The inference rules are organized into five groups corresponding to the five categories of inferred knowledge described in Subsection 2.1.

[4] Object and event information were extracted from the Sportec (`http://futbol.sportec.es`) and AS (`http://www.as.com/futbol`) portals respectively.

[5] `http://jena.sourceforge.net/`

3 Content Selection

3.1 Approach

The content selection module consists of a content bounding submodule and a content evaluation submodule. The content bounding submodule selects from the KB, using a set of hand-written rules, individuals that are relevant to the match for which a text is to be generated, either because they can be related directly to the match (e.g., the players of the teams involved, the match events such as goals), or because of the more general context of the competition (e.g., the league's classification). It also includes the logico-semantic relations that link these individuals. Given the large size (by NLG standards) of the KB, the motivation for the content bounder is to filter out irrelevant information and to make thus the subsequent content selection task more manageable. The output of the content bounder is a subset of the KB which constitutes the maximal set of data available for generating any sort of summary for a given match. The content structure presented in Figure 1 is a simplified output of the content bounding submodule.

The content evaluation submodule is in charge of evaluating the relevance of the content according to 1) a simple user model, 2) a set of heuristics, and 3) the logico-semantic relations that link individuals of the KB. Both the user model and the heuristics are numeric functions that map instances of concepts in the KB to a numeric measure of their relevance. The user model consists of the specification of the user's team of interest for the requested match or of a "neutral" profile—if the user has no favourite team. The heuristics measure relevance according to empirical knowledge extracted from a corpus of texts.[6] The content evaluation currently gives a weight of '1' if the node is related to the user's team of interest (or if the user profile is "neutral") and '0' otherwise. This weight is multiplied by the node's relevance measure, which is set to '1' if the heuristic weight for selecting the instance outweighs the heuristic weight for not selecting it. Otherwise it is set to '0'. Finally, the nodes that represent the logico-semantic relations are marked as relevant if they link two nodes with a positive relevance weight. This ensures the coherence of the content being selected. In Figure 1, given the user interest for the local team, the content selection heuristics and the logico-semantic relations, the five double circled nodes in the content structure are marked as relevant by the content evaluation submodule.

3.2 Empirical Determination of Relevance Measures

The weights of the instances that are to be selected are obtained by supervised training on a corpus of aligned data and online articles. The corpus consists of eight seasons of the Spanish League, from 2002/2003 to 2009/2010 with a total of 3040 matches, downloaded from different web sources. The articles typically consist of a title, a summary and a body. The data for each match consist of

[6] Relevance could also be measured according to other sources (e.g., past interaction with the user).

the teams, stadium, referee, players, major actions like goals, substitutions, red and yellow cards, and some statistical information such as number of penalties. Table [1] shows the verbalization of some categories in each of the three article sections considered for a single season in any of the sources. As can be seen, the result of the match (whether nominal or numerical) is almost always included in all the sections, whilst the verbalization of other categories is more extensive in the article body than in the summary, and in the summary more extensive than in the title. In our work on the generation of summaries, we focused on learning weights for league classifications, goals and red cards.

Table 1. Verbalization of some categories in title, summary and body of Spanish Football League articles (2007/2008 season) in all sources

	result	classification	goal	red card	stadium	referee	substitution
title	92.4%	16.3%	19.6%	9.3%	19.2%	2.9%	0%
summary	90.8%	22%	43.6%	32.2%	38.2%	3.7%	0.17%
body	97.6%	51.3%	95.2%	77.1%	82.4%	80%	18.1%

The data-text alignment procedure implies as a first step a preprocessing phase that includes tokenization and number-to-digit conversion. Then, instances of the relevant categories (i.e., specific goals, specific red cards, etc.) are detected using data anchors in the text (such as player names and team names) and regular expressions patterns compiled from the most frequent N word sequences of the corpus (where $1<N<5$). Data anchors are given priority over the use of regular expressions.

For the description of a goal or a red card, we used the same set of over 100 features. The features include deltas of minutes between the current event and the previous/next event of the same class, players and teams, information about individual players, a player's team and its classification. For modeling the classification, we used a more systematic approach to feature extraction by regarding a team's classification as event of a specific gameweek, comparing it to the events of the previous gameweek—that is, to the 20 classifications[7] of the previous gameweek and to the events of the same gameweek (also 20 classifications), such as the delta of category, points and team between classifications. In this way, we obtained a total of 760 features.

In order to classify the data, we used Boostexter [17], a boosting algorithm that uses decision stumps over several iterations and that has already been used in previous works on training content selection classifiers [1,10].[8] For each of the three categories (goal, red card, classification), we experimented with 15 different classifiers. We considered a section dimension (title, summary and title+summary) and a source dimension (espn, marca, terra, any one of them (any) and at least two of them), dividing the corpus each time into 90-10% of the matches for training and testing.

[7] The Spanish League competition involves 20 teams.

[8] After a number of experiments, the number of iterations was set to 300.

4 Discourse Structuring

The discourse structuring module receives as input a content plan which is a subset of the KB determined by the content bounding task, with some nodes marked as relevant by the content evaluation task (cf. Section 3). This subset of the KB in OWL-format is converted into the graph representation used by the Mate linguistic generation environment [2]. We use Mate for several reasons: 1) it comes with a handy API for graph manipulation, 2) it provides a straightforward representation of groups of nodes (i.e., "bubbles") necessary to represent discourse units, 3) the graph structures can be viewed in the Mate editor, and 4) the output of the discourse structuring is the input to the linguistic generation which uses these graph structures.

The discourse structuring works on the logico-semantic relations marked as relevant by the content selection (and their arguments, which are also relevant). It consists of three tasks which are: mapping logico-semantic to discourse relations, determining discourse units, and discourse units ordering. As already pointed out in [20] (for a different domain), a logico-semantic relation can be mapped onto different discourse relations depending on the user's previous knowledge, the content being communicated and the information structure. In the current prototype, the mapping between logico-semantic and discourse relations is one-to-one. The arguments of the logico-semantic relations are mapped onto nucleus–satellite arguments of the discourse relations following the Rhetorical Structure Theory [13]. For example, the CAUSE relation is mapped onto a *VolitionalCause* discourse relation, whilst the IMPLICATION relation is mapped onto a *NonVolitionalCause* discourse relation. As a consequence, the CAUSE relation between a set of goals and a victory can be verbalized during the linguistic generation by the discourse marker *gracias a* 'thanks to', whilst the IMPLICATION relation between a red card and an expulsion by *por* 'because of'.

The discourse unit determination is template-based; that is, we use our expertise of what can be said together in the same proposition in a football match summary. Currently, we have defined eleven discourse unit templates that cover the types of propositions that can be found in football summaries. Each core node, i.e., node that can be the argument of a discourse relation, can form a discourse unit. So, for each core node, a list of (possibly recursive) paths in the form *edge>Vertex* (where the edge is the object property and the vertex is the class range) is given to find in the graph the list of nodes that can be included in the discourse unit of that core node, starting from the core node. For example, the following is an excerpt of the template for expressing the result of a match:

```
partido>Partido,
periodo>PeriodoPartido,
resultNom>ResultNom,
resultNom>ResultNom>ganador>Equipo,
resultNom>ResultNom>perdedor>Equipo,
resultNom>ResultNom>protagonist>Equipo
```

This template includes the node *Partido* 'Match' when talking about the result, such that a sentence that introduces the match between the two teams and the final result (for example, nominal result with/without a numerical score) of the following kind can be produced: *The match between team A and team B ended with the victory of team A (2-1).* Any node that stays outside the discourse units is not included in the discourse plan. In other words, the discourse unit determination is in charge of further – fine-grained – content selection.

The final discourse structuring task, namely discourse unit ordering, consists of a simple partial order on the discourse units that starts with 'ResultPuntual'. In Figure 1, the simplified discourse plan consists of three ordered discourse units, each of which includes (double-circled) node(s) marked as relevant by the content evaluation submodule and further content nodes added by the discourse unit determination templates.

5 Evaluation

5.1 Content Selection Evaluation

Our evaluation of the content selection consisted of three stages: (1) evaluation of the automatic data-article alignment procedure, (2) evaluation of the performance of the classifiers for the empirical relevance determination, and (3) the evaluation of the content selection proper.

The evaluation of the automatic alignment against 158 manually aligned summaries resulted in an F-score of 100% for red cards, 87% for goals and 51% for classification. The low performance of classification alignment is due to the low efficiency of its anchors: positions, zones and points are seldom mentioned explicitly and both team names often appear in the summary, leading to ambiguity. For this reason, classification alignment was edited manually.

Table 2 shows the performance of the classifiers for the determination of the relevance of the three categories (goal, red card and classification) with respect to their inclusion into the summary section, comparing it to the baseline, which is the majority class. For red cards, the classifier did not show any significant improvement over the baseline for summary section in any of the cases involving summary section only. However, when considering title and summary from a source together, the classifier accuracy for red cards is 85% and the baseline 53% with $t = 4.4869$ ($p<0.0001$, sample size=62). In all cases, the best performance is obtained by considering the content from any of the online sources.

The evaluation of the content selection proper includes the template-based content selection performed during the discourse unit determination. The evaluation is done by comparing the content of generated summaries with that of existing summaries (the gold standard).

Our test corpus consists of 36 randomly selected matches from the set of matches of the 2007–2008 season, each with three associated summaries from three different web sources (namely espn, marca, terra). We compiled a list of all RDF-triples considered for inclusion in the content selection and discourse unit determination modules, including the logico-semantic relations. For each of the

Table 2. Performance of the best classifiers (vs majority baseline) on a test set for the summary section

category	source	sample size	classifier	baseline	paired t-test
goal	any	1123	64%	51%	t = 6.3360 ($p<0.0001$)
	terra	1121	65%	59%	t = 3.4769 ($p=0.0005$)
card	any	54	78.1%	65.4%	t = 1.6593 ($p=0.1030$)
classif	any	295	75%	61%	t = 4.4846 ($p<0.0001$)

108 (36×3) summaries, we manually annotated whether a triple was verbalized or not. We also annotated for each text the team of interest by checking whether the majority of content units was from one team or another; in case of equality, the user profile was considered neutral. This allowed us to compare the generated text of a given match for a given profile with the text(s) for the same profile. As baseline, we always select both teams and the final result regardless of profile since the result (and most likely the associated teams—as shown in Table 1) is almost always included in the summaries. This baseline is likely to have high precision and lower recall.

We performed three runs of generation: (1) a full run with relevance weights determined by the trained models ("estimated"), (2) a run in which the relevance of the instances is determined from the aligned texts, taking the profile into account ("real w., prof."), and (3) a run like (2), but without taking into account the user profile when determining relevance ("real w., no prof."). Table 3 shows the results of the evaluation for each of the three sources. Precision and recall are obtained by measuring the triples selected by the estimated or baseline model against the triples in the gold standard. The recall is predictably lower in the baseline than in the other runs. The F-measure in the source Marca is considerably lower for the three runs than the baseline. This is because the summaries in this source are very much like short titles (for marca, we had an average of 2 triples per summary vs. 4 for espn and 6 for terra). The runs without profile consideration have a somewhat lower F-measure than those with profile, especially for the two sources with the longest summaries. This shows that considering the profile of the user when selecting content is an important criterion. Finally, the performance of content selection with empirically estimated relevance is comparable to the performance of content selection with relevance taken from the target texts—which indicates that there are benefits in using supervised learning for estimating relevance.

Although a more formal error analysis would be needed, here are a few issues that we encountered during the (manual) counting of the triples for the evaluation:

1. errors in the automatic alignment for goals and red cards;
2. errors in the KB (we found at least a missing instance, and an error in the final score which meant that it was a draw instead of a victory);

Table 3. Content selection evaluation results

source	#triples	baseline			estimated			real w., prof.			real w., no prof.		
		prec.	rec.	F1	prec.	rec.	F1	prec.	rec.	F1	prec.	rec.	F1
espn	157	83.3	57.3	67.9	43.2	77.1	55.4	42.5	79.6	55.4	35.1	85.4	49.7
marca	74	49.0	63.5	55.3	21.8	79.7	34.2	20.2	79.7	32.2	17.7	90.5	29.6
terra	223	98.1	47.5	64.0	54.2	64.1	58.7	56.1	65.9	60.6	44.8	75.8	56.3

3. some inferred triples are missing, among them sets of goals for a given player or a given period of the match (e.g., first half) as well as some relations (e.g., violation of expectation between the fact that team A did not win and team B played with less than 11 players during a determined period of the game);
4. some of the considered triples are never included in the final content plan; for instance, the sets of goals without the listing of the individual goals (to say that a team marked 3 goals).

With respect to the second issue, we would like to point out that although we did not evaluate the correctness of the KB, we are aware that it is not error-free and that more testing and mending is needed. With respect to the third and fourth issues, the question comes up how to systematize the discovery of new inferred knowledge (including relations) and how to get relevance heuristics for content selection. Supervised learning can be unreliable and/or painstaking, especially if the data is scarce and/or requires manual annotation. Another promising avenue of research is to obtain those heuristics from the user using reinforcement learning.

5.2 Discourse Structuring Evaluation

To evaluate the coherence of the final texts, we relied on the evaluation of their *readability* done on 51 matches with three different outputs, one for each of the three user profiles (team A, team B and Neutral) performed by ten evaluators external to the project. As pointed out by [14]: "Fluency concerns the quality of generated text, rather than the extent to which in conveys the desired information. This is related to the notion of 'readability' and will include notions such a syntactic correctness, stylistic appropriateness, **organization and coherence**." *(the emphasis is ours)*

Figure 4 shows the questionnaire on readability passed to the evaluators. The questionnaire consists of a five point scale. We asked the evaluators not to judge the content of the texts as such, but rather their structure (and grammaticality). For each text, we obtained a total of three different judgements. These judgements were averaged to give the text its final score. We obtained an average performance for readability of 88%, which is indicative of the high degree of coherence of the texts.

Please select one of the following:

5 The text is very easy to read; it seems perfectly natural.
4 The text is easy to read although there are some details that seem unnatural.
3 The text is not too difficult to read, but there are annoying repetitions or abusive agglutination of information in the same sentence.
2 The text is difficult to read, due to the reasons above, but it's still worth an effort.
1 The text is not readable.

Fig. 4. The readability questionnaire put to the subjects

6 Related Work

Natural Language Generation systems generally use hand-crafted toy knowledge bases (KBs) and/or external databases (DBs) as input. Sometimes, data is assessed or evaluated in order to produce new inferred knowledge that is more suitable for being communicated in natural language texts [16,20]. From the DBs/KBs, it is extracted for inclusion in schemas or discourse plan operators [9]. A few systems reason directly on the input representation [5,4]. In [5], a *content potential* is constructed based on the domain model (museum artifacts) that consists of predicates between entities, related by discourse relations. The content selection approach consists in weighting the relevant facts from a given node based on a number of manually set criteria and opportunistically navigating the best (and closest) facts. In [4], a content graph is hand-crafted that includes redundancy relations between facts. The nodes in the graph are weighted according to the PageRank centrality formula, with the best ranked nodes selected.

Some work has also been done on empirically estimating the relevance of content using supervised learning in the bibliographical domain [6] and the sports domain [1,10].

In recent years, there has been also a surge of interest in using OWL knowledge bases for Natural Language Generation (NLG) [3,8,15,21]. These works have mainly focused on verbalizing the taxonomic content of ontologies and/or annotating them with linguistic information for linguistic realization. None, to the best of our knowledge, is dedicated to text planning, be it content selection or discourse structure

7 Conclusions and Future Work

We have presented an application-independent two-layer OWL ontology and a text planning approach that exploits it. During our work, we have faced two typical problems faced by NLG practitioners when massaging content into text: the mapping between world knowledge and domain communication knowledge [11], and the mapping between world knowledge and linguistic knowledge. The problem related to the first type of mapping was resolved by adding to the basic ontology a second layer populated using inference rules. The problem related to the second type of mapping was resolved by grouping the content nodes into discourse units, which are then mapped onto a (near-standardized) conceptual structure that can be used by linguistic generation. Our content selection works directly on the evaluation of the relevance of the nodes in the ontology based on empirically obtained relevance weights, a simple user profile and the conceptualized logico-semantic relations instantiated in the KB. The discourse structuring module consists of three well-defined albeit somewhat basic tasks. The evaluation shows that the user profile is an important criterion when selecting content for this domain, and that empirical determination of the relevance is a viable approach.

In the medium-term, we would like to make the tasks of our content selection and discourse structuring modules domain-independent, that is, parametrizable to a given domain but with clearly domain independent mechanisms. This is being addressed by applying the approach for ontology-based content selection to a completely different domain, namely environmental information. Thus, we want to be able to bound the content using a general algorithm that exploits domain-specific specifications of what content is to be bound. Similarly, the mapping operations we have developed for mapping the discourse structure to the conceptual structure should be general, although the actual mapped units are domain-dependent. We also need to develop a set of general purpose content extraction algorithms (such as PageRank [4]) that are applied once the content has been evaluated. We are furthermore working on the implementation of a constraint-based discourse structuring approach to replace the template-based discourse unit ordering task. Additional work is projected on the discourse unit determination, as it is still somewhat dependent on the ordering with respect to the information structure (what to say first where), and, finally on a mapping formalism between logico-semantic and discourse relations that shall include rules that take into account the user's previous knowledge, the content, etc.

The context for achieving our goal is optimal in that we can draw, on the one hand, upon the formalism and tools for inferencing provided by OWL (and thus count on standardized solutions for tasks prior to text planning) and, on the other hand, upon a theoretically motivated and mature linguistic generator, with clearly defined representations of the linguistic description (and thus count on standardized solutions for tasks coming after text planning).

References

1. Barzilay, R., Lapata, M.: Collective content selection for concept-to-text generation. In: Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing (2005)
2. Bohnet, B., Wanner, L.: Open Source Graph Transducer Interpreter and Grammar Development Environment. In: Proceedings of the Seventh Conference on International Language Resources and Evaluation, LREC 2010 (2010)
3. Bontcheva, K., Wilks, Y.: Automatic Report Generation from Ontologies: The MIAKT Approach. In: Meziane, F., Métais, E. (eds.) NLDB 2004. LNCS, vol. 3136, pp. 324–335. Springer, Heidelberg (2004)

4. Demir, S., Carberry, S., McCoy, K.F.: A Discourse-Aware Graph-Based Content-Selection Framework. In: Proceedings of the International Natural Language Generation Conference (2010)
5. O'Donnell, M., Mellish, C., Oberlander, J., Knott, A.: ILEX: An architecture for a dynamic hypertext generation system. Natural Language Engineering 7, 225–250 (2001)
6. Duboue, P.A., McKeown, K.R.: Statistical Acquisition of Content Selection Rules for Natural Language Generation. In: Proceedings of Empirical Methods for Natural Language Processing (EMNLP), pp. 121–128 (2003)
7. Dukle, K.: A Prototype Query-Answering Engine Using Semantic Reasoning. Master Thesis. University of South Carolina (2003)
8. Galanis, D., Androutsopoulos, I.: Generating Multilingual Descriptions from Linguistically Annotated OWL Ontologies: the NaturalOWL System. In: Proceedings of the 11th European Workshop on Natural Language Generation (ENLG 2007). Schloss Dagstuhl, Germany (2007)
9. Hovy, E.H.: Automated Discourse Generation Using Discourse Structure Relations. Artificial Intelligence 63(1-2), 341–386 (1993)
10. Kelly, C., Copestake, A., Karamanis, N.: Investigating content selection for language generation using machine learning. In: Proceedings of the 12th European Workshop on Natural Language Generation, pp. 130–137 (2009)
11. Kittredge, R., Korelsky, T., Rambow, O.: On the need for domain communication knowledge. Computational Intelligence 7(4), 305–314 (1991)
12. Lenat, D.B.: CYC: a large-scale investment in knowledge infrastructure. Communications of the ACM 38(11), 33–38 (1995)
13. Mann, W., Thompson, S.: Rhetorical Structure Theory: Toward a functional thoery of text organization. Text 8(3) (1988)
14. Mellish, C., Dale, R.: Evaluation in the Context of Natural Language Generation. Computer Speech and Language 12, 349–373 (1998)
15. Mellish, C., Pan, J.: Natural Language Directed Inference from Ontologies. Artificial Intelligence 172(10), 1285–1315 (2008)
16. Reiter, E.: An Architecture for Data-to-Text Systems. In: Proceedings of ENLG 2007, pp. 97–104 (2007)
17. Schapire, R.E., Singer, Y.: BoosTexter: A boosting-based system for text categorization. Machine Learning 39(2/3), 135–168 (2000)
18. Tsinaraki, C., Polydoros, K.F., Christodoulakis, S.: Ontology-based semantic indexing for mpeg-7 and tv-anytime audiovisual content. Multimedia Tools and Applications 26(3), 299–325 (2005)
19. Uschold, M., King, M.: Towards a Methodology for Building Ontologies. In: Proceedings of the Workshop on Basic Ontological Issues in Knowledge Sharing, Held in Conduction with IJCAI 1995, pp. 6.1–6.10 (1995)
20. Wanner, L., Bohnet, B., Bouayad-Agha, N., Lareau, F., Nicklass, D.: MARQUIS: Generation of User-Tailored Multilingual Air Quality Bulletins. Applied Artificial Intelligence 24(10), 914–952 (2010)
21. Wilcock, G.: Talking owls: Towards an ontology verbalizer. In: Proceedings of the Human Language Technology for the Semantic Web and Web Services, ISWC 2003, Sanibel Island, Florida, pp. 109–112 (2003)

Linking Lexical Resources and Ontologies on the Semantic Web with Lemon

John McCrae, Dennis Spohr, and Philipp Cimiano

AG Semantic Computing, CITEC, University of Bielefeld
{jmccrae,dspohr,cimiano}@cit-ec.uni-bielefeld.de

Abstract. There are a large number of ontologies currently available on the Semantic Web. However, in order to exploit them within natural language processing applications, more linguistic information than can be represented in current Semantic Web standards is required. Further, there are a large number of lexical resources available representing a wealth of linguistic information, but this data exists in various formats and is difficult to link to ontologies and other resources. We present a model we call *lemon* (Lexicon Model for Ontologies) that supports the sharing of terminological and lexicon resources on the Semantic Web as well as their linking to the existing semantic representations provided by ontologies. We demonstrate that *lemon* can succinctly represent existing lexical resources and in combination with standard NLP tools we can easily generate new lexica for domain ontologies according to the *lemon* model. We demonstrate that by combining generated and existing lexica we can collaboratively develop rich lexical descriptions of ontology entities. We also show that the adoption of Semantic Web standards can provide added value for lexicon models by supporting a rich axiomatization of linguistic categories that can be used to constrain the usage of the model and to perform consistency checks.

1 Introduction

The Semantic Web has made available a large amount of semantic data in the form of ontologies and there have been several attempts to apply this to NLP tasks such as question answering [17], information extraction [7] and text generation [2]. However, current standards such as RDFS and SKOS [18] only allow for limited linguistic information to be attached to an ontology, limiting the potential functionality of these applications. In contrast, there are a large number of rich sources of linguistic information that have been created including term bases, lexica (e.g., Lefff [20]) and machine readable dictionaries (e.g., WordNet [9]). However, much of this data is confined by the format and distribution methodology to "data silos" and as such cannot be easily shared or extended. This proves to be a specific disadvantage for the creation of lexical resources for specific domains (e.g., SNOMED [22]), as these terminologies inevitably need to reuse basic terms from general-domain resources. For these reasons, we propose a new model called *lemon* (Lexicon Model for Ontologies) that is designed to

© Springer-Verlag Berlin Heidelberg 2011

allow lexical information to be represented relative to an ontology and shared on the Semantic Web. The *lemon* model has the following crucial features: i) it represents a concise and thus reusable model, ii) it is based on RDF(S), iii) it is "open" in the sense that it does not prescribe the usage of a particular inventory of linguistic categories and properties, but instead iv) supports the reuse of any linguistic ontology such as GOLD [8] or ISOcat [13], and v) assigns semantics to lexical entries by way of reference to ontological entities in line with Buitelaar [4].

There have already been several attempts to define linguistic ontologies, notably the GOLD ontology [8] and the OLiA ontologies [5]. However, these ontologies are primarily focused on providing specific linguistic categories and do not define a methodology for representing morphosyntactic information. Instead, we base our model on the Lexical Markup Framework (LMF) [10] with the goal of making lexica interoperable. In particular, we include the idea of *data categories* [19], which are uniquely identified concepts that can be used for computational linguistic tasks, such as those compiled by the ISOCat [13] project.

The paper is structured as follows: after discussing related work in Section 2, in Section 3 we introduce the *lemon* model as a basic model for representing linguistic information relative to ontologies, that uses existing ontologies and/or data category registries to represent specific linguistic categories. In Section 4 we present an extension of *lemon* called *lemon*-LexInfo which makes particular choices with respect to the linguistic categories and properties that can be modelled by importing categories from ISOCat and COMLEX [16], for instance. In Section 5 we present three experiments which show how *lemon* lexica can be created automatically as well as by reuse of WordNet. In a first experiment we show that legacy lexica such as WordNet can be easily converted to the *lemon* format. The main benefit here is that by this move, lexica can be linked to each other, extended and reused in a straightforward manner by exploiting the RDF datamodel and the Linked Data principles [1]. In a second experiment, we show how *lemon*-lexica for already existing ontologies can be created in an automatic fashion by building on standard NLP components, thus substantially reducing the costs of creating such lexica. Finally, in a third experiment, we show how general lexica such as WordNet can be reused when constructing a lexicon for a specific vocabulary such as FOAF, thus saving costs and resources in the creation of *lemon*-lexica. We conclude in Section 6.

2 Background

RDFS's label property provides a simple way to attach a lexical form to an ontological concept. The SKOS model [18] goes further by allowing to define a preference order on labels as "preferred", "alternative" and "hidden." However, modern lexica as developed in the lexical resources community, for example Lefff [20], contain more information than can be succinctly represented with these vocabularies, in particular morphology, phrase structure and subcategorization information. WordNet [9] is of course one of the most well-known lexica and there have been several attempts to adapt it to the Semantic Web, e.g. by transforming it into an RDF format [24] and "ontologizing" it [12]. However, these lexica are limited by the actual amount of data available in WordNet and by the format of WordNet itself. In addition, the conceptual model used by WordNet has been identified as unsound from an ontological persepective (see [11]). In general, we wish for a model that is capable of representing a large variety of linguistic information and can do this for an arbitrary ontology.

While SKOS fails on the former, WordNet and similar domain-independent resources fail with respect to the latter. These two desiderata can be met by building on standardisation efforts carried out in the lexical resources community, in particular the Lexical Markup Framework [10]. LMF is capable of representing a wide variety of linguistic information, however it has no mechanism for relating lexica to ontologies and instead relies on a traditional word sense model as in WordNet, which has been criticised by Kilgariff[14].

The LexInfo model [6] is an ontology-lexicon model which has a clearly separate linguistic layer and a semantic-syntactic correspondence object. The LexInfo model was created by importing LMF. But the authors noted there were many technical issues with this, not least that there is still no canonical form of LMF that is usable for the Semantic Web, in the sense of being correct RDF and having dereferencable URIs. The authors fixed this by publishing their own version of LMF[1] and enhancing it by introducing names for the property relations: i.e., replacing LMF's 3 original properties (`isAssociated`,`isPartOf`,`isAdorned`) with more specific links such as `hasWordForm`.

Table 1. Number of values for part of speech in some existing formalisms

	WordNet (2.0)	GOLD	ISOcat
Number of values	5	81	115

	OLiA	LexInfo (1.0)	
Number of values	174	11	

An important problem in the representation of lexica is that there is significant disagreement between different models with respect to the properties and values of linguistic annotation that are needed to represent a lexicon. Take as an example the case of part of speech, which is a property that most lexica represent but often have a significant disagreement with respect to the number and granularity of part-of-speech tags. For example, in Table 1 we show the number of values of part of speech that can be represented by some language resource schemas. As we can see, the scope of the representation varies greatly. In addition, there is some disagreement about what constitutes a part of speech: for example, GOLD [8] considers "comparative adjective" to be a part of speech value, while the other formalisms consider the "comparative" value to be assigned to a property "degree." Furthermore, there is also disagreement about the hierarchy of these concepts: for example, GOLD has no class for "Verb" and instead groups adjectives, adverbs and verbs under the concept "predicator".

[1] `http://www.lexinfo.net/lmf`

The OLiA project [5] attempted to solve this problem by aligning several linguistic annotation ontologies including GOLD, and ISOcat to a reference ontology. Within the ISO TC 37, the problem of different linguistic annotations has been handled by data categories, which are sets of values for such properties. These are currently being collected by the ISOcat project [13]. For our purposes, we require a formalism that does not distinguish between these different resources and can use any of them depending on the wishes of the lexicon creator.

3 The *lemon* Model

We present the *lemon* model – illustrated in Figure 1 – as our proposal for a lexicon model for ontologies. The *lemon* model consists of a lexicon object with a number of (lexical) entries. Each of these entries can then be further described with morphosyntactic properties, and mapped via (lexical) sense objects to entities in the ontology. The core elements of the ontology are as follows

– **Lexicon:** The lexicon is realised as a resource. Each lexicon is mono-lingual and is marked with a language tag and optionally a topic
  • *Example:* A lexicon may consist of English names for diseases.
– **Lexical Entry:** The lexical entry represents a single term within the lexicon. As morphosyntactic information is attached to the lexical entry, each entry must have the same syntax. Hence term variants, such as abbreviation,

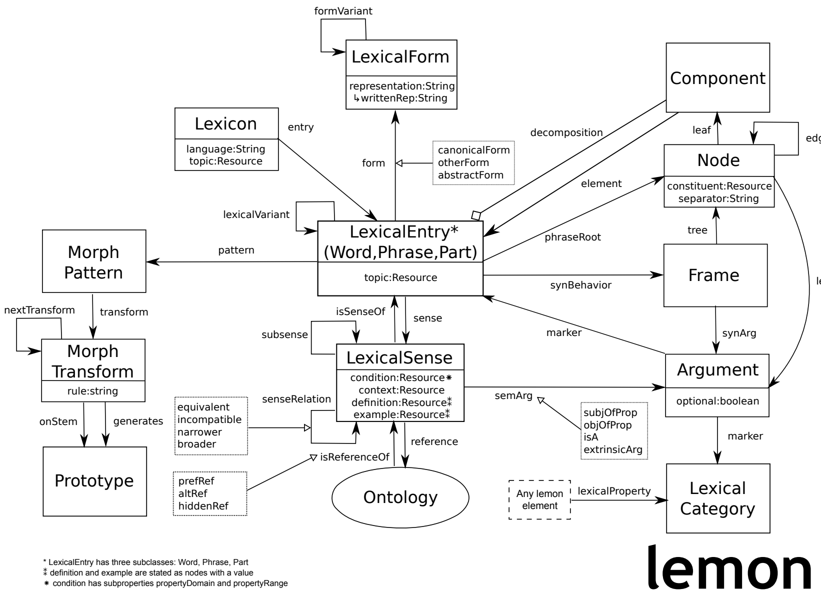
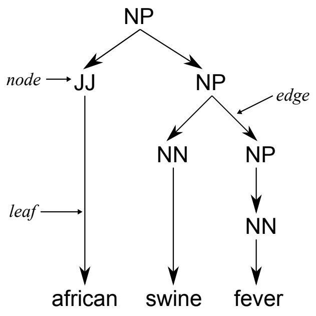


Fig. 1. The *lemon* model

are represented as separate lexical entries and marked as **lexicalVariant**s. Lexical entries are split up into three subclasses: **Word**, **Phrase** and **Part** (of word).
  - *Example:* "Cancer of the mouth" is a lexical entry in the lexicon. "Mouth cancer" would be another lexical entry, marked as a lexical variant of the first.
- **Form:** Each lexical entry consists of a number of forms. These represent different inflectional variants of the entry and may be marked as **canonical** (lemma), **other** or **abstract**.
  - *Example:* The lexical entry for "bacterium", may have two forms: the canonical "bacterium" and the other form "bacteria".
- **Representation:** Each form may have multiple representations, of which the most important is the written representation, but other representations such as a phonetic form are also possible.
  - *Example:* The written representation of the form of bacterium would be "bacterium". It may also have a phonetic representation bktim
- **Lexical Sense:** Unlike in other models, *lemon*'s senses are not assumed to be finite or disjoint. Instead, the sense represents the correspondence between the lexical entry and the ontology entity. It may include extra specification of this correspondence such as context and condition, or human-readable annotations such as definitions or examples. It may be indicated as the **preferred**, **alternative** or **hidden** lexicalisation of an ontology entity, by analogy to the preference order on labels defined by SKOS [18] in terms of *preferred*, *alternative* and *hidden*.
  - *Example:* The lexical entries for "influenza" and "flu" may both refer to an ontology entity `http://purl.org/obo/owl/DOID#DOID_8469` (in the OBO foundry ontologies [21]). Each entry would have a separate sense object. The former sense would be marked as used in a scientific context, and the latter as a layman term.
- **Reference:** The meanings of a lexical entry are specified through a "reference" to an ontology entity, and hence the lexical entry is linked to the semantic description given by the ontology.
- **Property:** Any element in a *lemon* model may be further described by a property. *lemon* offers a generic property `lexicalProperty`, which other linguistic properties should derive from, so that all lexical properties can be grouped.
  - *Example:* The forms, "bacterium" and "bacteria", may have a property "number" with values "singular" and "plural" respectively. The lexical entry may also be marked with part of speech noun.
- **Frame** and **Argument:** Subcategorization frames represent the valency of verbs and other lexical predicators, i.e., the number and type of arguments it can or should take. Each argument is also represented as a resource and is linked both from the frame, to indicate the syntactic role, and from the sense, to indicate the semantic role.
  - *Example:* For example, the property `complicated_by` may be represented by a lexical entry with a frame corresponding to "Y complicates X" where "X" and "Y" are the arguments of this frame.

– **Component:** Each lexical entry may be split up into a list of components, each of which refers to other lexical entries. This is used for showing which words compose a multi-word expression or a compound word. The list of components are stated as an RDF list, hence the list of components is ordered and finite.
  - *Example:* The German term "hmorrhagisches Fieber" ("haemorrhagic fever"), is composed of two components "hmorrhagisch" and "Fieber." The first component may have properties to indicate that it is the form with neuter adjectival agreement. Decompositions may also be used with compound words, for example the German term, "Ebolavirus" ("Ebola virus"), may have a decomposition into "Ebola" and "Virus."
– **Node:** Each lexical entry may be associated with a phrase structure. This consists of a number of nodes linked by either **edge** or **leaf** arcs to components
  - *Example:* A parse tree may be constructed for the term "African swine fever" as below.



This is useful as it indicates that this term is understood as an "African" version of "swine fever" instead of a "fever" affecting "African swine."

The *lemon* model thus provides a general framework by which we can represent lexica linked to ontologies to specify the semantics of lexical entries. However, for most applications a specific vocabulary needs to be introduced to describe the specific linguistic categories used in the model. We do not have space to go into the full details of the usage of the model. A full technical report on the *lemon* model is available at `http://www.lexinfo.net/lemon-cookbook.pdf`.

It should be noted that *lemon* is not technically an instantiation of LMF as there are many differences in the modelling of semantics and optimizations due to the adoption of RDF. However, many aspects of *lemon* do correspond directly to LMF and in fact there is a *lemon*-LMF converter available at `http://www.lexinfo.net/lemon2lmf`.

In order to represent information such as part of speech, we can include this information by referencing URIs from a data category registry. GOLD, OLiA and ISOcat all provide URI based identifiers for their systems, such that we can reference any property and gain more information about this annotation by deferencing this URI. As *lemon* is based on RDF, it is trivial to include these URIs as resources in our lexicon scheme. For example, the following represents a single lexical entry for the Dutch word "maag" ("stomach").

```
@prefix lemon: <http://www.monnet-project.eu/lemon#> .
@prefix isocat: <http://www.isocat.org/datcat/> .

:maag
   lemon:canonicalForm [ lemon:writtenRep "maag"@nl ;
        isocat:DC-1298 isocat:DC-1387 ] ; # number=singular
   lemon:otherForm     [ lemon:writtenRep "magen"@nl ;
        isocat:DC-1298 isocat:DC-1354 ] ; # number=plural
   isocat:DC-1345 isocat:DC-1333 ; # partOfSpeech=noun
   isocat:DC-1297 isocat:DC-1880 ; # gender=feminine
   lemon:sense [
     lemon:reference <http://purl.org/obo/owl/EHDAA#EHDAA_2993> ] .

isocat:DC-1298 rdfs:subPropertyOf lemon:property .
isocat:DC-1345 rdfs:subPropertyOf lemon:property .
isocat:DC-1297 rdfs:subPropertyOf lemon:property .
```

Note that the prescribed URIs for ISOcat data categories are specified with a registration number. For legibility, we include comments to give a human-readable description of the properties used in the example.

Expanding on this example we can also model multilinguality by including lexical entries that have the same reference in different languages, for example:

```
:maag
  lemon:canonicalForm [ lemon:writtenRep "maag"@nl ] ;
  lemon:sense [
    lemon:reference <http://purl.org/obo/owl/EHDAA#EHDAA_2993> ] .

:stomach
  lemon:canonicalForm [ lemon:writtenRep "stomach"@en ] ;
  lemon:sense [
    lemon:reference <http://purl.org/obo/owl/EHDAA#EHDAA_2993> ] .
```

In this way we can publish translated versions of common lexicon without any need to modify the original lexicon or the original ontology.

4 The *lemon*-LexInfo Model

lemon can accomodate any data category scheme, however the relations between the data categories and the *lemon* model must be specified in each lexicon. For this reason we adapted the LexInfo model[6], which was originally introduced as an extension of LMF. This second version of LexInfo was engineered from the ground up using *lemon* and importing data categories from the ISOcat data category registry. For *lemon*-LexInfo we expand on the RDF schema used in *lemon* with OWL to create links to these external linguistic ontologies as well as to axiomatize certain linguistic types, subcategorization frames in particular and to further constrain their meaning and usage, thus supporting consistency checks.

- **Axiomatic definitions of linguistic categories:** From ISOcat [13] we converted the DCIF files for the morphosyntax section into RDF and imported them into the *lemon*-LexInfo model. We mapped the "complex" data categories to RDF properties and the "simple" data categories to RDF resources (OWL Individuals). This gave us a very large set of properties that can be used to describe entries in the lexicon. We then defined each of these properties as subproperties of the appropriate *lemon* property. Most of these came under the general **lexicalProperty** arc, but some were mapped elsewhere, e.g., **register** was modelled as a (sense) **context**, as it represents a semantic distinction on the usage of a term.
- **Instantiating a hierachy of categories:** For each of the properties adopted from ISOcat, their range was defined in terms of a class having as individuals all elements in the extension of this class according to ISOcat DCIF. In this way we can introduce hierarchy among the annotations, e.g., **properNoun** and **commonNoun** are both members of the classes **PartOfSpeech** and **NounPOS**. *lemon*'s three lexical entry classes (**Word**, **Phrase** and **Part**) were further subclassed into specific classes, e.g, **Verb**, **NounPhrase**. As each of these classes could be related to the properties introduced from ISOcat, we introduced appropriate axioms to define these classes. For example:

$$Noun \equiv \exists partOfSpeech.NounPOS$$

- **Compositional definition of subcategorization frames:** We define linguistic frames in a precise manner in terms of the sets of arguments they have. We introduced a set of syntactic role properties, for example, "subject", "object" and then created precise OWL definitions of a each frame from the COMLEX [16] vocabulary.
  We can now define an intransitive frame as a frame with a subject, no direct object and no indirect object as follows:

$$IntransitiveFrame \equiv (= 1 subject) \sqcap (= 0 directObject) \sqcap (= 0 indirectObject)$$

  We found that we could further simplify the description of subcategorization frames by defining abstract frames such as **PrepositionalObjectFrame**, so we could then define a hierarchy of frames. E.g.,

$$PrepositionalObjectFrame \equiv \exists prepositionalObject$$

$$IntransitivePPFrame \equiv IntransitiveFrame \sqcap PrepositionalObjectFrame$$

  In this way, we reduced COMLEX's 163 frames to 36 basic frames and 4 modifiers to describe argument control. These are listed at `http://www.lexinfo.net/basic-frames`.

This illustrates the value of using Semantic Web standards, as we did not need to define specific vocabulary to define these linguistic concepts. Instead, OWL was sufficient to provide powerful modelling of linguistic concepts. To further illustrate this point, we note that it is also then possible to use OWL to define linguistic conditions, such as "every French noun is masculine and/or feminine," without requiring an extra modelling language such as in LMF.

5 Experiments

5.1 Converting WordNet to *lemon*

As we wish to use *lemon* to create Semantic Web lexica, we require that it is capable of representing legacy lexical resources. One of the largest, freely available lexica is WordNet [9] and it seems clear that it is necessary for *lemon* to be able to represent the information in this resource. We based our work on the existing RDF version of WordNet [24] and then simply aligned this to the *lemon* model. We proceeded as described below to yield a *lemon*-compatible version of WordNet.

Methodology

– We mapped WordNet's synsets to *lemon*'s references. This means that the synsets and the links between them form a quasi-ontology and replace the role that the ontology plays in normal usage of *lemon*, i.e. assigning meaning to lexical entries by reference to ontological entitites. The advantage of this separation is that we can introduce mappings to more sound semantic models such as OntoWordNet [12] without affecting the original data.
– The definition of word sense in WordNet and *lemon* corresponded well, as WordNet's word senses can be defined as the sub-meaning of a word belonging to a particular synset and *lemon*'s sense as the intersection between the lexical usage of the entry and the semantic usage of the ontology entity.
– We also found that the definition of word in WordNet and word in *lemon* corresponded, so mapped these appropriately. We note here that the original RDF version of WordNet actually loses information as alternative forms are listed in the original WordNet format. Hence, we manually extracted them and added them to the *lemon* representation.
– Finally, WordNet marks part of speech on the sense and synset level, whereas *lemon* does it on the word level. We switch the properties to the lexical entries using the morphosyntactic properties of LexInfo that were originally derived from ISOcat.

As such a brief example of a rewritten WordNet synset is as follows (`lwn` is the lemon-WordNet namespace and `wn20` the original WordNet-RDF mapping):

```
lwn:marmoset-noun-entry rdf:type lemon:LexicalEntry ;
  lexinfo:partOfSpeech lexinfo:noun ;
  lemon:sense lwn:sense-marmoset-noun-1 ;
  lemon:canonicalForm lwn:word-marmoset-canonicalForm .

lwn:sense-marmoset-noun-1 lemon:reference wn20:synset-marmoset-noun-1 .

lwn:word-marmoset-canonicalForm lemon:writtenRep "Marmoset"@en .
```

Discussion. We found that the *lemon* model was relatively close to WordNet. By mapping this to a common vocabulary, we believe it should make it easier to combine multiple lexica without losing information. In addition, as *lemon* can provide more complex representations of syntactic and morphological information we believe this could enable WordNet to be further extended vertically. In the future, we intend to extend this work by incorporating other open-source lexica, such as Wiktionary[2] and Lefff [20]. These resources are available at `http://www.lexinfo.net`.

5.2 Generating *lemon*-LexInfo Models

The main goal of *lemon* is to create lexica that can be used to describe ontologies, as such, for the second experiment we chose to create a model for a widely used ontology. The "Friend of a friend" (FOAF) [3] is an ideal candidate for testing whether the model is concise and there is a large amount of FOAF data available on the Web. As the model is small, it seems feasible to develop a corresponding lexicon manually in order to guarantee a high quality result.

Methodology. We used the *lemon*-LexInfo lexicon generation service available at `http://monnetproject.deri.ie/Lemon-Editor`, which provides an interface for working with *lemon* lexica and incorporates a number of basic NLP tasks so that it can auto-generate most of the information required for lexicon generation. In particular, the service has the following features:

- Extraction of labels from RDFS, SKOS or URI fragments.
- Tokenization yields sub-components. In particular, we used the standard tokenizer that is packaged with the Lucene information retrieval library[3].
- Part of speech tagging to give simple morphosyntactic features. We used the Stanford Tagger [23] for our experiments.
- Lemmatization to identify which forms are canonical. We again used the Stanford Tagger to perform this.
- Parsing to produce phrase structure. For this, we used the Stanford Parser [15].
- Subcategorization identification using a rule-based system. Each rule consists of the following:
  - A phrase structure pattern to detect the structure of the label, represented by the lexical entry. For example, the pattern, `FRAG (VP, PP)`, indicates a fragment consisting of a verb phrase and a prepositional phrase, such as "located in".
  - A set of a classes which the ontology entity should be an instance of (by RDF's `type` property). These are generally basic OWL types such as `ObjectProperty`.
  - The class of the generated frame.
  - The definition of the arguments of the frame based on the syntactic and semantic roles it has.

[2] `http://www.wiktionary.org/`

[3] Available at `http://lucene.apache.org`

Table 2. Results by component for lexicon generation on the FOAF ontology

	Total Errors	Total Correct	Precision
Tokenizer	1	112	99.1%
Tagger	8	105	92.9%
Parser	12	101	89.4%
Subcategorizer	6	57	92.1%
Subcategorizer (with parses corrected)	2	61	96.8%
Total	21	92	81.5%

We used the service by uploading a standard version of the FOAF ontology. This version of FOAF contained 63 entities (of which 12 were classes and 51 properties) and the generation process created 113 Lexical Entries (note that extra lexical entries were created to describe multiple word expressions). The results by component are described in Table 2. The results show that the lexicon generation is very accurate at different levels having accuracy levels between 89.4% (parsing of labels) and 99.1% (tokenization of labels). Overall, the number of lexical entries for which there is an error with respect to one level of linguistic analysis is 21 out of 103, thus corresponding to an overall accuracy of 81.5%. This is a very satisfactory result showing that we can generate lexica for a given ontology effectively and efficiently.

Discussion. The tokenizer component was quite accurate producing only one error (splitting "E-commerce" into two words), and the tagger was relatively accurate. Most of the tagger's errors were related to not distinguishing correctly between common nouns and proper nouns. The parser was responsible for most of the errors, in particular the implementation we used was biased to produce full sentence parses. For example, the label "work info homepage" was interpreted as an imperative sentence instead of a noun phrase[4]. The subcategorizer was generally correct, however, it should be noted that the vast majority of the labels in the source ontology (like with many ontologies) are simply noun phrases with the result that the subcategorization frames were mostly noun phrases with possessive adjunct (i.e., "X is the homepage of Y") or unary noun phrase predicates (i.e., "X is a homepage"). Once we corrected all the incorrect parses and reran the subcategorizer, the accuracy improved, generating only two incorrect frames: "X is the Myers Briggs of Y" and not recognizing "account" in "holds account" as the object of the verb. We would hope to conduct a more thorough evaluation of this component in later work, on an ontology with more complex labels and modelling for predicates with arity greater than two (e.g., donative structures).

[4] Discarding sentence parses was not effective here as the next best parse was the same verb phrase.

5.3 Merging Generated Lexica with Existing LR

Obviously, a large amount of the terminology used within the FOAF vocabulary is also found within WordNet. Thus, it seems that it would be advantageous to reuse the WordNet entries when defining a lexicon for FOAF. We can easily achieve this in *lemon* by creating a sense object for each meaning specified in the FOAF ontology and then linking it to our *lemon*-aligned version of WordNet if possible and only creating a new lexical entry if this is not possible, thus fostering reuse, producing more compact lexica and ultimately reducing the costs in lexicon creation.

Methodology. We took the WordNet lexicon generated by the approach described in Section 5.1 and the lexicon for the FOAF ontology generated in section 5.2, and compared each of the entries in the two lexica. We used the following criteria to evaluate if two entries were equivalent:

– The written representation of the canonical from was the same or differed only by capitalization of the initial letter.
– The part of speech tag was equal, if specified.
– The two entries did not have a linguistic property with different values. Note that we still counted the entries as equal if one entry did not have a value for the linguistic property.
– The non-canonical forms could be matched in such a way that each corresponding pair had the same written representation and did not have contradictory property values. We need to search for similar pairs here as it is possible that one lexicon may have, for example, “made” as both the preterite and past participle form of the entry “make”.

Table 3. Number of lexical entries for FOAF lexicon mapped to WordNet

	Number	Percentage
Mapped to WordNet	78	69.0%
Not mapped (MWE)	25	22.1%
Not mapped (Proper Noun)	9	7.9%
Not mapped (other)	1	0.9%

We then used this definition of equality to map FOAF lexical entries to WordNet, replacing the generated FOAF lexical entries with WordNet entries whenever an equivalent – as defined above – WordNet entry exists. Note that we only mapped to the words within the WordNet model and not to senses. Thus, for an ambigous term like “ID” we did not decide whether the meaning in FOAF was as an abbreviation of “identification” or “Idaho.” We note here that *lemon* does not require us to make this distinction, but we can as it is possible to reuse either lexical entries, which aren’t semantically disambiguated, or senses, which are disambiguated. The results of this mapping process are depicted in Table 3. The table shows that we can successfully map 69% of the lexical entries derived from the FOAF ontology to appropriate WordNet lexical entries. The remaining 31% of the cases can be broken down into three groups: firstly, there were a number of multiple word expressions in FOAF that were not contained within WordNet, for example "past project" or "online gaming account." Secondly, FOAF contained a number of proper nouns to refer to specific social networking services such as "MSN" ("The Microsoft Network") or AIM ("AOL Instant Messenger") although some proper nouns were contained within WordNet, e.g., "Yahoo." Thirdly, we found one neologism, "weblog", that was not in WordNet.

Discussion. These results show that there is significant value in reusing existing lexical resources in the creation of lexica for new domains, as the majority of terms used by the FOAF ontology were also found in WordNet. However, we also conclude that most domain ontologies will need to introduce new terminology, and as such there is a necessity to collaboratively expand lexical resources, through the use of linked data and semantic web search engines. In particular, this is most notable for multiple word expressions and proper nouns, both of which are contained in WordNet, but only to a limited degree.

6 Conclusion

We have presented our model *lemon*, which acts as a basic model for publishing lexica on the Semantic Web and connecting them to ontologies. The model's openness allows it to be a concise model and hence easy to use and work with. We have also introduced an extension of the model called *lemon*-LexInfo which makes specific design choices by reusing existing linguistic categories defined in ISOcat and COMLEX. The reuse of existing data categories exemplifies how *lemon* can be used to publish lexical resources in a way that avoids the data being confined to "silos." We have demonstrated that the RDF based foundations of *lemon* make it trivial to include these data categories. Furthermore, the use of RDF allows us to gain added value in the description of our lexica, as we demonstrate by using OWL to simplify the process of describing subcategorization frames. By converting WordNet to *lemon*, we demonstrate the utility of *lemon* as an interchange format that could be used to bring complementary lexical resources together under a single framework. In this way, we believe this model could be used to bring together lexical resources with the semantic modelling on the Semantic Web. We also show that by the use of standard NLP components we can generate high-quality lexica. Finally, we show that for developing lexical resources for specific domains both the reuse of existing lexical resources and the generation of new lexical resources can be used together to effectively and collaboratively develop new resources.

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. Bontcheva, K.: Generating tailored textual summaries from ontologies. In: Gómez-Pérez, A., Euzenat, J. (eds.) ESWC 2005. LNCS, vol. 3532, pp. 531–545. Springer, Heidelberg (2005)
3. Brickley, D., Miller, L.: FOAF Vocabulary Specification 0.98 (2010) (accessed December 3, 2010)
4. Buitelaar, P.: Ontology-based Semantic Lexicons: Mapping between Terms and Object Descriptions. In: Ontology and the Lexicon, pp. 212–223. Cambridge University Press, Cambridge (2010)
5. Chiarcos, C.: Grounding an Ontology of Linguistic Annotations in the Data Category Registry. In: Proceedings of the 2010 International Conference on Language Resource and Evaluation, LREC (2010)
6. Cimiano, P., Buitelaar, P., McCrae, J., Sintek, M.: LexInfo: A Declarative Model for the Lexicon-Ontology Interface. Web Semantics: Science, Services and Agents on the World Wide Web 9(1), 29–51 (2011)
7. Collier, N., Doan, S., Kawazoe, A., Goodwin, R., Conway, M., Tateno, Y., Ngo, Q., Dien, D., Kawtrakul, A., Takeuchi, K., Shigematsu, M., Taniguchi, K.: BioCaster: detecting public health rumors with a Web-based text mining system. Oxford Bioinformatics 24(24), 2940–2941 (2008)
8. Farrar, S., Langendoen, D.: Markup and the GOLD Ontology. In: Proceedings of Workshop on Digitizing and Annotating Text and Field Recordings (2003)
9. Fellbaum, C.: WordNet: An electronic lexical database. MIT Press, Cambridge (1998)
10. Francopoulo, G., George, M., Calzolari, N., Monachini, M., Bel, N., Pet, M., Soria, C.: Lexical markup framework (LMF). In: Proceedings of the Fifth International Conference on Language Resource and Evaluation (LREC 2006) (2006)
11. Gangemi, A., Guarino, N., Masolo, C., Oltramari, A.: Sweetening wordnet with DOLCE. AI Magazine 24(3), 13 (2003)
12. Gangemi, A., Navigli, R., Velardi, P.: The ontoWordNet project: Extension and axiomatization of conceptual relations in wordNet. In: Chung, S., Schmidt, D.C. (eds.) CoopIS 2003, DOA 2003, and ODBASE 2003. LNCS, vol. 2888, pp. 820–838. Springer, Heidelberg (2003)
13. Kemps-Snijders, M., Windhouwer, M., Wittenburg, P., Wright, S.: ISOcat: Corralling data categories in the wild. In: Proceedings of the 2008 International Conference on Language Resource and Evaluation, LREC (2008)
14. Kilgarriff, A.: I Don't Believe in Word Senses. Computers and the Humanities 31(2), 91–113 (1997)
15. Klein, D., Manning, C.: Accurate unlexicalized parsing. In: Proceedings of the 41st Annual Meeting on Association for Computational Linguistics, pp. 423–430 (2003)
16. Korhonen, A., Krymolowski, Y., Briscoe, T.: A large subcategorization lexicon for natural language processing applications. In: Proceedings of the Fifth International Conference on Language Resources and Evaluation, LREC 2006 (2006)
17. Lopez, V., Pasin, M., Motta, E.: AquaLog: An ontology-portable question answering system for the semantic web. In: Gómez-Pérez, A., Euzenat, J. (eds.) ESWC 2005. LNCS, vol. 3532, pp. 546–562. Springer, Heidelberg (2005)

18. Miles, A., Bechhofer, S.: SKOS Simple Knowledge Organization System Reference (2009) (accessed October 19, 2010)
19. Romary, L.: Standardization of the formal representation of lexical information for NLP. In: Dictionaries: An International Encyclopedia of Lexicography. Mouton de Gruyter, Berlin (2010)
20. Sagot, B.: The Lefff, a freely available and large coverage morphological and syntactic lexicon for French. In: Proceedings of the 2010 International Conference on Language Resource and Evaluation (LREC) (2010)
21. Smith, B., Ashburner, M., Rosse, C., Bard, J., Bug, W., Ceusters, W., Goldberg, L., Eilbeck, K., Ireland, A., Mungall, C., et al.: The OBO Foundry: coordinated evolution of ontologies to support biomedical data integration. Nature Biotechnology 25(11), 1251–1255 (2007)
22. Spackman, K., Campbell, K., Côté, R.: SNOMED RT: a reference terminology for health care. In: Proceedings of the AMIA Annual Fall Symposium, pp. 640–644 (1997)
23. Toutanova, K., Klein, D., Manning, C., Singer, Y.: Feature-rich part-of-speech tagging with a cyclic dependency network. In: Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology, pp. 173–180 (2003)
24. Van Assem, M., Gangemi, A., Schreiber, G.: Conversion of WordNet to a standard RDF/OWL representation. In: Proceedings of the Fifth International Conference on Language Resources and Evaluation, LREC 2006 (2006)

Elimination of Redundancy in Ontologies

Stephan Grimm and Jens Wissmann

FZI – Research Center for Information Technology, Karlsruhe, Germany
{grimm,wissmann}@fzi.de

Abstract. Ontologies may contain redundancy in terms of axioms that logically follow from other axioms and that could be removed for the sake of consolidation and conciseness without changing the overall meaning. In this paper, we investigate methods for removing such redundancy from ontologies. We define notions around redundancy and discuss typical cases of redundancy and their relation to ontology engineering and evolution. We provide methods to compute irredundant ontologies both indirectly by calculating justifications, and directly by utilising a hitting set tree algorithm and module extraction techniques for optimization. Moreover, we report on experimental results on removing redundancy from existing ontologies available on the Web.

1 Introduction

Ontologies are subject to an engineering lifecycle as any other technical artifact in an information system, and methods for their development and maintenance over time are researched under the label of *ontology evolution*. Ontologies also provide features for automated deduction that allow for deriving implicit knowledge from its explicitly stated axioms. These features give rise to a notion of redundancy in ontologies, meaning the presence of axioms that implicitly follow from other axioms present in the ontology. One important technique in an ontology evolution toolbox is the automated identification and elimination of such redundancy on demand to provide a means for the consolidation and compactification of ontologies in the course of their development.

Although there might be scenarios where redundancy can arguably be a desirable feature, we consider cases in which ontology engineers want to identify and eliminate redundancy to consolidate and clean up their ontologies for the sake of maintenance. Techniques for keeping ontologies irredundant have many use-cases in ontology evolution scenarios as redundancy can cause various problems. Due to the non-locality of redundant information distributed over several axioms in different places, an ontology can be hard to understand, to maintain and to be split into separate independent modules. Moreover, deletion or update of axioms might be semantically ineffective in case they are implied by others. Furthermore, in scenarios where techniques of automated knowledge acquisition are utilized, redundancy elimination can reduce the amount of acquired statements for optimising local storage and reasoning. Similarly, it can be used for undoing materialization.

© Springer-Verlag Berlin Heidelberg 2011

Elimination of redundancy can also support other techniques for processing ontologies. While module extraction techniques typically single out relevant parts of an ontology for reuse, they do not ensure minimality in axioms, whereas a combination with redundancy elimination allows for obtaining irredundant modules. Another use case for irredundant ontologies is the handling of access rights to axioms of an ontology. There, the problem of concealed axioms being derived by accessible ones can be avoided when using irredundant ontologies.

In this paper, we study methods for the elimination of redundancy in OWL DL ontologies. We define various notions around redundancy in the axioms of an ontology and identify typical cases in which redundancy is introduced unanticipatedly in the course of an ontology's evolution over time. We provide two methods for computing all irredundant versions of an ontology: one by means of calculating justifications combined with internalization, and one by direct identification of redundant axioms in an optimized hitting-set-tree algorithm that makes calls to an underlying description logic reasoner. Furthermore, we present empirical results for eliminating redundancy in existing ontologies based on an implementation of the above methods, where we report on both the efficiency of the algorithms and the effect of redundancy elimination applied to prevalent ontologies. By this, we contribute an effective and efficient means for consolidating ontologies on demand to any ontology engineering toolbox.

The paper is organized in sections. After recalling preliminaries and related work in Section 2, we introduce our notions around redundancy in ontologies and discuss typical cases of redundancy in Section 3. Then, we describe methods for an automated elimination of redundancy in Section 4 before reporting on experimental results in Section 5. We conclude with future work in Section 6.

2 Preliminaries

We introduce some basic notions around ontologies and review related work.

OWL and Description Logics. As a language for ontologies we consider the prominent Web Ontology Language (OWL) [11], which is based on the description logic (DL) formalism [2]. In description logics, an *ontology* $\mathcal{O}$ is a set of axioms that express either terminological (T-Box) or assertional (A-Box) knowledge. For details about types of axioms and the way complex concepts are constructed from individual, concept and role names, we refer to [2]. Our results are largely independent from the concrete DL used.

Inference with OWL ontologies builds on the notion of logical consequence, and we write $\mathcal{O} \models \alpha$ to mean that an ontology $\mathcal{O}$ logically *entails* an axiom α, and $\mathcal{O} \models \{\alpha_1, \ldots, \alpha_n\}$ to express entailment of several axioms.

The *signature* of an ontology $\mathcal{O}$, denoted by $\sigma(\mathcal{O})$, is the set of all individual, concept and role names occurring in $\mathcal{O}$, and thus, its vocabulary.

Moreover, by $\langle\mathcal{O}\rangle$ we denote the *deductive closure* of an ontology $\mathcal{O}$, i.e. the set $\{\alpha \mid \mathcal{O} \models \alpha\}$ of all DL axioms α that are logical consequences of $\mathcal{O}$.

Internalization. A technique called *internalization* can be used to express an ontology in form of a single concept inclusion axiom [2]. The internalization of an ontology $\mathcal{O}$ results in an axiom $\alpha_{\mathcal{O}}$ that contains all semantic information in $\mathcal{O}$, i.e. $\langle\mathcal{O}\rangle = \langle\{\alpha_{\mathcal{O}}\}\rangle$. However, OWL ontologies cannot be fully internalized in their complete expressivity. Certain role axioms cannot be internalized as they do not syntactically fit the form of the concept inclusion axiom $\alpha_{\mathcal{O}}$ (see e.g. [8]).

Justifications. For debugging ontologies, *justifications* are used to provide explanations for entailments. A justification is a minimal subset of an ontology that supports a given entailment, captured by the following definition from [9].

Definition 1 (justification). *For an ontology $\mathcal{O}$ and an axiom α with $\mathcal{O} \models \alpha$, a set $\mathcal{J}$ of axioms is a* justification for α in $\mathcal{O}$ *if $\mathcal{J} \subseteq \mathcal{O}$, $\mathcal{J} \models \alpha$ and there is no set $\mathcal{J}'$ such that $\mathcal{J}' \subset \mathcal{J}$ and $\mathcal{J}' \models \alpha$.*

Module Extraction. Techniques of module extraction are used to obtain a fragment of an ontology that is semantically relevant for entailments over a given signature. Despite containing only a subset of the original ontology's axioms, a *module* preserves all entailments with regard to this signature. We adapt the definition of a module based on the notion of conservative extension from [4].

Definition 2 (module). *Let $\mathcal{O}$ and $\mathcal{O}_m$ be ontologies with $\mathcal{O}_m \subseteq \mathcal{O}$ and Σ be a signature. Then, $\mathcal{O}_m$ is a* module for Σ in $\mathcal{O}$ *if for every ontology $\mathcal{O}'$ with $\sigma(\mathcal{O}') \cap \sigma(\mathcal{O}) \subseteq \Sigma$, we have that $\mathcal{O}' \cup \mathcal{O} \models \alpha$ if and only if $\mathcal{O}' \cup \mathcal{O}_m \models \alpha$ for any axiom α with $\sigma(\alpha) \subseteq \Sigma$.*

Related Work. A different notion of redundancy used in the DL literature, e.g. [2], refers to concept names in ontologies that are entailed to be equivalent to others, and thus are redundant vocabulary. Our notion of redundancy in axioms has not been extensively investigated in the context of OWL ontologies or description logics, and we mainly build on ideas introduced in [10] about redundant clauses in propositional logic formulas. Moreover, our main method for eliminating redundancy is largely inspired by the work on the use of a hitting-set-tree algorithm [12] for finding all justifications in [9].

In various works on normal forms, DL knowledge bases are transformed into a more compact form, such as prime implicate normal form [3] or linkless concept descriptions [5]. In contrast to yielding irredundant ontologies, these works are targeted to pre-processing ontologies for more efficient reasoning.

In [6], reductions of RDFS ontolgies are investigated in terms of RDF graphs that do not contain entailed triples explicitly, and their uniqueness is related to subsumption acyclicity of the original RDF graphs. Due to the simpler semantics of RDFS, however, these results only cover explicit class subsumption.

In [1], the compactification of ontologies has more been studied more with an application in mind and less focused on a grounding in the formal semantics underlying OWL.

3 Redundancy in Ontologies

In many ontology evolution scenarios, redundancy is unconsciously introduced in the axioms of an ontology, which can make it hard to maintain. In the following, we introduce useful notions and discuss cases of redundancy in ontologies.

3.1 Notion of Redundancy

For an ontology it is often desirable to derive a minimal version that does not contain any redundancy in its axioms but has the same semantical “meaning”. An ontology contains redundancy in terms of expressed axioms if any of the axioms it contains is entailed by other axioms contained. The removal of such a redundant axiom would preserve the deductive closure of the ontology due to this entailment. Accordingly, we define *redundancy* in ontologies as follows.

Definition 3 (redundancy). *An ontology $\mathcal{O}$ is* redundant *if it contains an axiom α such that $\mathcal{O} \setminus \{\alpha\} \models \alpha$.*

We also call an irredundant subset of an ontology $\mathcal{O}$ that preserves the deductive closure of $\mathcal{O}$ a *reduction* of $\mathcal{O}$, as defined next.

Definition 4 (reduction). *Let $\mathcal{O}$ and $\hat{\mathcal{O}}$ be ontologies such that $\hat{\mathcal{O}} \subseteq \mathcal{O}$. Then, $\hat{\mathcal{O}}$ is a* reduction of $\mathcal{O}$ *if $\hat{\mathcal{O}}$ is irredundant and $\langle\hat{\mathcal{O}}\rangle = \langle\mathcal{O}\rangle$.*

In case an ontology is already irredundant, it is its own reduction. Hence, every ontology always has a reduction, which, however, need not be unique. For a focused removal of redundancy from parts of an ontology only, any part can be replaced by any of its reductions without loosing entailments due to monotonicity of DLs. For studying cases of an ontology having several reductions, we classify its axioms according to their level of *dispensability*, similar as in [10].

Definition 5 (dispensability of axioms). *For $\mathcal{O}$ an ontology and α an axiom,*

- *α is* indispensable in $\mathcal{O}$ *if it is contained in all reductions of $\mathcal{O}$*
- *α is* unconditionally dispensable in $\mathcal{O}$ *if it is in no reduction of $\mathcal{O}$;*
- *α is* conditionally dispensable in $\mathcal{O}$ *if there are two different reductions $\hat{\mathcal{O}}_1, \hat{\mathcal{O}}_2$ of $\mathcal{O}$ such that $\alpha \in \hat{\mathcal{O}}_1$ and $\alpha \notin \hat{\mathcal{O}}_2$.*

Indispensable axioms are those required in an ontology, unconditionally dispensable axioms those that can safely be removed, and conditionally dispensable axioms those that are interchangeably replaceable. Clearly, the existence of several reductions for an ontology is connected to the presence of conditionally dispensable axioms, as expressed in a proposition adapted from results in [10].

Proposition 1. *An ontology $\mathcal{O}$ has a unique reduction if and only if the following interchangeable conditions hold:*

1. *$\mathcal{O}$ has no conditionally dispensable axioms;*
2. *$\langle\mathcal{O}\rangle = \langle\mathcal{O}_i\rangle$ for $\mathcal{O}_i$ the axioms indispensable in $\mathcal{O}$;*
3. *there are no distinct sets S_1, S_2 of axioms conditionally dispensable in $\mathcal{O}$ such that $\langle\mathcal{O}\rangle = \langle\mathcal{O} \setminus S_1\rangle = \langle\mathcal{O} \setminus S_2\rangle \neq \langle\mathcal{O} \setminus (S_1 \cup S_2)\rangle$.*

Unfortunately, there is no easy way for distinguishing the conditionally dispensable axioms from the unconditionally dispensable ones. However, we can at least easily identify those axioms as unconditionally dispensable that follow from the indispensable axioms, according to the following lemma.

Lemma 1. *For an ontology $\mathcal{O}$ with indispensable axioms $\mathcal{O}_i$, any axiom $\alpha \in \mathcal{O}$ with $\mathcal{O}_i \models \alpha$ is unconditionally dispensable in $\mathcal{O}$, and thus, an ontology $\hat{\mathcal{O}}$ is a reduction of $\mathcal{O}$ if and only if it is a reduction of $\mathcal{O}_i \cup \{\alpha \in \mathcal{O} \mid \mathcal{O}_i \not\models \alpha\}$.*

A special case of redundancy covered by Lemma 1 is the presence of tautologies. Obviously, any tautological axiom in an ontology is unconditionally dispensable, and thus, no reduction can contain a tautology.

For the computation of reductions, we will be interested in a gradual removal of dispensable axioms from an ontology one-by-one. Removing single dispensable axioms from an ontology results in a strict decrease of the set of dispensable axioms in the respective sub-ontologies, such that reductions do not cease to be reductions by such removal, as stated in the following lemma.

Lemma 2. *Let $\mathcal{O}$ be an ontology with dispensable axioms $\mathcal{O}_d$ and α be an axiom with $\alpha \in \mathcal{O}_d$. Then, $\mathcal{O}'_d \subset \mathcal{O}_d$ for $\mathcal{O}'_d$ the axioms dispensable in $\mathcal{O} \setminus \{\alpha\}$, and thus, any reduction of $\mathcal{O} \setminus \{\alpha\}$ is also a reduction of $\mathcal{O}$.*

3.2 Cases of Redundancy

Since we look at redundancy in ontologies primarily from an ontology engineering and evolution point of view, we are interested in cases of redundancy as they occur in an ontology, and in the way redundancy is introduced in such cases.

Ontologies typically evolve over time with their different parts being developed and maintained in different contexts and separately from each other, which is a potential source for redundancy if such parts are combined. Moreover, big ontology development projects involve multiple ontology engineers who might introduce redundancy in the course of concurrent modeling activities.

Table 1 shows an example of a redundant ontology $\mathcal{O}_{hum}$ about human relationships that illustrates the notions introduced above. The axioms in Table 1 are listed in the order they were introduced, starting with basic notions about humans being either male or female, having children and being mother or father. These are followed by the notion of parent and more precise definitions of what it means to be father or mother, reflecting the successive expansion of an ontology in its evolution process. Overall, $\mathcal{O}_{hum}$ has two reductions and occurrences of both dispensable and indispensable axioms.

We identify cases in which redundancy is introduced in an unanticipated way.

Definition Over Subsumption. Definitions of concept names bear the potential to overwrite previously introduced subsumption axioms that involve these names, as they often succeed the introduction of new concepts by simple subsumption axioms. Axioms like $A \sqsubseteq B_1, A \sqsubseteq B_2$ become dispensable when adding a definition $A \equiv B_1 \sqcap B_2$ for A due to $\{A \equiv B_1 \sqcap B_2\} \models \{A \sqsubseteq B_1, A \sqsubseteq B_2\}$. In our example from Table 1, this case applies to the entailment $\{(10)\} \models (8)$.

Table 1. An example ontology about human relationships

$\mathcal{O}_{hum}$		
(1)	$Human \sqsubseteq \exists\, hasChild^{-}.Human$	humans are children of humans
(2)	$Human \equiv Male \sqcup Female$	humans are defined as either male or female
(3)	$Male \sqcap Female \sqsubseteq \bot$	males and females are distinct
(4)	$Father \sqsubseteq Human$	fathers are human
(5)	$Mother \sqsubseteq Human$	mothers are human
(6)	$Father \sqcap Mother \sqsubseteq \bot$	fathers and mothers are distinct
(7)	$Parent \equiv Human \sqcap \exists\, hasChild.Human$	parents are just humans with human children
(8)	$Father \sqsubseteq Parent$	fathers are parents
(9)	$Human \sqsubseteq = 2\, hasChild^{-}.Human$	humans have exactly two human parents
(10)	$Father \equiv Male \sqcap Parent$	fathers are male parents
(11)	$Mother \equiv Female \sqcap Parent$	mothers are female parents
(12)	$Parent(Peter)$	Peter is a parent
(13)	$Male(Peter)$	Peter is male
(14)	$Father(Peter)$	Peter is a father

indispensable:{2,3,7,9,10,11}, conditionally disp.: {12,13,14}, uncond. disp.:{1,4,5,6,8}
reductions for $\mathcal{O}_{hum}$: $\hat{\mathcal{O}}_1$ = {2,3,7,9,10,11,14}, $\hat{\mathcal{O}}_2$ = {2,3,7,9,10,11,13,12}

Subsumption Transitivity. Also simple subsumption axioms between concept names can introduce redundancy when explicating transitive parts of the subsumption hierarchy, e.g. by introducing additional intermediate subclasses. If $A \sqsubseteq C$ is stated about A first, and later on the intermediate subclass B is introduced by means of $A \sqsubseteq B$, $B \sqsubseteq C$ then $A \sqsubseteq C$ becomes dispensable. In our example from Table 1, this is the case for the entailment $\{(7), (8)\} \models (4)$.

Conjunctive Strengthening. Simple subsumption axioms between concept names bear the potential to be overwritten by more complex ones that strengthen the restrictions on the subsumed class by means of conjunctions. When placing new concept names in an ontology's subsumption hierarchy by means of axioms like $A \sqsubseteq B_1$, and then adding more restrictions by means of conjunction, such as $A \sqsubseteq B_1 \sqcap B_2$, at a later stage, the former axiom becomes dispensable. The entailment $\{(10)\} \models (8)$ with regard to Table 1 is also an example for this case.

Disjunctive Weakening. Disjunctions on the right-hand side of subsumption axioms bear the potential to become redundant when some of the disjunctive cases are excluded later on. An example are so called coverage axioms of the form $A \sqsubseteq B_1 \sqcup B_2$, which are a common design pattern and thus likely to be introduced early in the evolution of an ontology. If at a later stage more restrictive subsumptions about A are added without replacing the original coverage axiom, such as $A \sqsubseteq B_1$, then this one becomes dispensable due to $A \sqsubseteq B_1 \models A \sqsubseteq B_1 \sqcup B_2$.

Cardinality Inclusion. Also numbers in cardinality restrictions might introduce redundancy when applied to the same role but at different places or stages. The restriction $A \sqsubseteq \leq 3\ r$, for example, is overwritten by $A \sqsubseteq \leq 2\ r$, due to $A \sqsubseteq \leq 2\ r \models A \sqsubseteq \leq 3\ r$. For our example from Table 1, we get e.g. $\{(9)\} \models (1)$.

Inherited Disjointness. Subclasses of respective disjoint classes are again disjoint from each other. Hence, disjointness axioms introduced deep down in class hierarchies become dispensable when the respective parent classes are stated to be disjoint. In our example from Table 1, we have that $\{(3)\} \models (6)$.

Assertional Redundancy. Also concept or role assertion axioms in ABoxes can be redundant in combination with TBox information. This might either happen in case a knowledge engineer describes a situation in an ABox very explicitly, not having the full information regarding all TBox axioms in mind that derive part of this situation, or in case a TBox is extended after situations have been described accurately in the ABox in a way that makes ABox statements obsolete. In our example from Table 1, axiom (14) is equivalent to $\{(12), (13)\}$.

For our experiments with OWL ontologies, we focus on scenarios in which ontologies are primarily hand-crafted and no techniques of automated knowledge acquisition are used. In such scenarios, we assume "typical" domain ontologies as used in the Semantic Web to contain rather little redundancy, since ontology engineers would most likely model their ontologies in a rather concise way, not repeating statements over and over in different ways. Therefore, we work with the following hypothesis for the subsequent computation of reductions.[1]

Hypothesis 1 (low redundancy). *An ontology is expected to contain significantly less dispensable axioms $\mathcal{O}_d$ than indispensable axioms $\mathcal{O}_i$: $\#\mathcal{O}_d \ll \#\mathcal{O}_i$.*

4 Computing Reductions

Starting from the notion of redundancy introduced above, we investigate how irredundant ontologies can be produced from redundant ones in an automated way. We propose two methods for computing reductions, one that builds on both internalization and the computation of justifications, and one that computes reductions directly based on the hitting set algorithm for diagnosis problems.

4.1 Finding Reductions by Computing Justifications

Recall the notion of a justification, which is a minimal subset of axioms of an ontology that support a particular entailment. As such, a justification has the property of not containing any redundancy at the level of expressed axioms, since excluding any of its axioms would give up the entailment. If we expand a single entailment to cover the whole ontology, we can extend this property from the restricted set of axioms in the justification to the ontology as a whole. We can do so by using the mechanism of internalization in order to encode all information of the original ontology in a single entailed axiom – giving up this entailment amounts to loosing information with regard to the original ontology. Hence, algorithms for computing justifications can be used to produce reductions.

Since there can be several justifications for an entailment, this approach results in several solutions for eliminating redundancy in an ontology. In fact, algorithms that compute all justifications produce all reductions of an ontology internalized by the respective entailed axiom, as stated next.

[1] In Section 5 we will support this hypothesis with some empirical evidence from experiments with existing ontologies.

Algorithm 1. computeRedJust ($\mathcal{O}$; R) – Compute reductions via justifications

```
Require: an ontology $\mathcal{O}$
Ensure: $R$ is the set of all reductions of $\mathcal{O}$
  $\mathcal{O}' := R := J := \emptyset$
  extractInternalisablePart($\mathcal{O}$; $\mathcal{O}'$)
  internalize($\mathcal{O}'$; $\alpha_{\mathcal{O}'}$)
  computeJustifications($\mathcal{O}$, $\alpha_{\mathcal{O}'}$; $J$)
  for all $J_i \in J$ do
    $R := R \cup \{J_i \cup (\mathcal{O} \setminus \mathcal{O}')\}$
  end for
```

Theorem 1. *Let $\mathcal{O}$ be an ontology and $\alpha_{\mathcal{O}}$ be the internalization of $\mathcal{O}$ such that $\langle\mathcal{O}\rangle = \langle\{\alpha_{\mathcal{O}}\}\rangle$. Any justification $\mathcal{J}_{\mathcal{O}}$ for $\alpha_{\mathcal{O}}$ in $\mathcal{O}$ is a reduction of $\mathcal{O}$ such that $\langle\mathcal{O}\rangle = \langle\mathcal{J}_{\mathcal{O}}\rangle$.*

Proof. Let $\mathcal{J}_{\mathcal{O}}$ be a justification for $\alpha_{\mathcal{O}}$ in $\mathcal{O}$. Then, the entailment $\mathcal{J}_{\mathcal{O}} \models \alpha_{\mathcal{O}}$ holds, and since $\mathcal{O}$ and $\alpha_{\mathcal{O}}$ are semantically equivalent, also $\mathcal{J}_{\mathcal{O}} \models \mathcal{O}$. Moreover, due to $\mathcal{J}_{\mathcal{O}} \subseteq \mathcal{O}$ we also have that $\mathcal{O} \models \mathcal{J}_{\mathcal{O}}$. This implies $\langle\mathcal{O}\rangle = \langle\mathcal{J}_{\mathcal{O}}\rangle$.

We have just seen that $\alpha_{\mathcal{O}} \in \langle\mathcal{J}_{\mathcal{O}}\rangle$ due to $\mathcal{J}_{\mathcal{O}} \models \alpha_{\mathcal{O}}$. Since $\mathcal{J}_{\mathcal{O}}$ is a minimal set of axioms that entail $\alpha_{\mathcal{O}}$, we get that $\mathcal{J}' \not\models \alpha_{\mathcal{O}}$ for any $\mathcal{J}' \subset \mathcal{J}_{\mathcal{O}}$. Hence, we get $\langle\mathcal{J}'\rangle \neq \langle\mathcal{J}_{\mathcal{O}}\rangle$ for any $\mathcal{J}' \subset \mathcal{J}_{\mathcal{O}}$, and by Definition 3, $\mathcal{J}_{\mathcal{O}}$ is irredundant. □

One drawback of this method is that in OWL not all axioms can in general be internalized, and axioms not covered, such as transitivity or other role properties, are not taken into account for the computation of reductions. However, for cases in which ontologies are expressed in a language fragment that contains such axioms, at least the internalizable part of an ontology can be freed of redundancy. Algorithm 1 provides a procedure for this that makes use of procedures for internalization and computation of justifications like the ones in [9].[2] By this, it provides a way to eliminate redundancy in ontologies by means of the readily available techniques for internalization and for computing justifications.

Another drawback of this method is its low efficiency. Internalization of the whole ontology in an axiom to be checked for entailment doubles the input for the underlying DL reasoner, which has a significant impact on the run time of the typically exponential time DL reasoning problem. Therefore, we investigate the direct computation of reductions, next.

4.2 Finding Reductions by Direct Diagnosis

A straightforward method for computing a single reduction of an ontology is to successively remove dispensable axioms from the ontology, which requires linearly many entailment checks for verifying dispensability of single axioms in the successive sub-ontologies. In case the ontology's dispensable axioms have been

[2] In our procedural notation, parameters before the ;-symbol are read-only and passed by value, while those after the ;-symbol are read/write and passed by reference.

Algorithm 2. computeSingleReduction ($\mathcal{O}$, $\mathcal{O}_d$; $\hat{\mathcal{O}}$) – Compute a single reduction for an ontology with pre-identified dispensable axioms

```
Require: an ontology $\mathcal{O}$, the set $\mathcal{O}_d$ of axioms dispensable in $\mathcal{O}$
Ensure: $\hat{\mathcal{O}}$ is a reduction of $\mathcal{O}$
  $\hat{\mathcal{O}} := \mathcal{O}$
  for all $\alpha \in \mathcal{O}_d$ do
    if $\hat{\mathcal{O}} \setminus \{\alpha\} \models \alpha$ then $\hat{\mathcal{O}} := \hat{\mathcal{O}} \setminus \{\alpha\}$
  end for
```

Algorithm 3. determineDispensableAxioms ($\mathcal{O}$, $\mathcal{O}^*$; $\mathcal{O}_d$) – Find dispensable axioms

```
Require: an ontology $\mathcal{O}$, an ontology $\mathcal{O}^*$ with $\mathcal{O}^* \subseteq \mathcal{O}$ and $\mathcal{O}^*$ contains all axioms
  dispensable in $\mathcal{O}$
Ensure: $\mathcal{O}_d$ is the set of axioms that are dispensable in $\mathcal{O}$
  $\mathcal{O}_d := \emptyset$
  for all $\alpha \in \mathcal{O}^*$ do
    if $\mathcal{O} \setminus \{\alpha\} \models \alpha$ then $\mathcal{O}_d := \mathcal{O}_d \cup \{\alpha\}$
  end for
```

pre-identified, this can be optimized verifying only dispensable axioms, since indispensable axioms do not need to be checked for removal, according to Definition 5. Algorithm 2 provides a procedure for computing a single reduction of an ontology with possibly pre-identified dispensable axioms in its second parameter; in case of not knowing the dispensable axioms in advance, this parameter is initialized with the whole ontology.

For the task of finding all (or more than one) reductions of an ontology, it is beneficial to spend the effort of pre-identifying dispensable axioms, since for all reductions to be computed the indispensable axioms can be neglected, and due to Hypotheses 1 their relative number is expected to be high. Algorithm 3 provides a procedure for this pre-identification, taking as its second parameter a subset of the original ontology for optimization in case some axioms can be excluded from being dispensable, which applies for repeated calls when only a subset of formerly dispensable axioms needs to be checked due to Lemma 2.

Based on pre-identified dispensable axioms, techniques for ontology module extraction can be applied to optimize the computation of all reductions of an ontology. Observe from Algorithm 2 that calls to a DL reasoner for entailment checking are restricted to axioms dispensable in the original ontology, which can expected to be few according to Hypothesis 1. Instead of checking entailment with respect to the full ontology, it is therefore sufficient to only take into account an ontology module that is computed by means of the signature of all dispensable axioms to preserve their (non-)entailment according to Definition 2. The following proposition provides the basis for this optimization.

Proposition 2. *Let $\mathcal{O}$ be an ontology with dispensable axioms $\mathcal{O}_d$ and $\mathcal{O}_m$ be a module for $\sigma(\mathcal{O}_d)$ in $\mathcal{O} \setminus \mathcal{O}_d$. Then, $\hat{\mathcal{O}}' \cup (\mathcal{O} \setminus (\mathcal{O}_m \cup \mathcal{O}_d)$ is a reduction of $\mathcal{O}$ for any reduction $\hat{\mathcal{O}}'$ of $\mathcal{O}_m \cup \mathcal{O}_d$.*

Proof. $\hat{\mathcal{O}}' \cup (\mathcal{O} \setminus (\mathcal{O}_m \cup \mathcal{O}_d)) \subseteq \mathcal{O}$ holds due to $\hat{\mathcal{O}}' \subseteq \mathcal{O}_m \cup \mathcal{O}_d \subseteq \mathcal{O}$.

Since $\hat{\mathcal{O}}'$ is a reduction of $\mathcal{O}_m \cup \mathcal{O}_d$, we have that $\langle \hat{\mathcal{O}}' \rangle = \langle \mathcal{O}_m \cup \mathcal{O}_d \rangle$, and thus, $\langle \hat{\mathcal{O}}' \cup (\mathcal{O} \setminus (\mathcal{O}_m \cup \mathcal{O}_d)) \rangle = \langle (\mathcal{O}_m \cup \mathcal{O}_d) \cup (\mathcal{O} \setminus (\mathcal{O}_m \cup \mathcal{O}_d)) \rangle = \langle \mathcal{O} \rangle$.

Finally, assume that $\hat{\mathcal{O}}' \cup (\mathcal{O} \setminus (\mathcal{O}_m \cup \mathcal{O}_d))$ is redundant. Then, there is an axiom $\alpha \in \hat{\mathcal{O}}' \cup (\mathcal{O} \setminus (\mathcal{O}_m \cup \mathcal{O}_d))$ with $\hat{\mathcal{O}}' \cup (\mathcal{O} \setminus (\mathcal{O}_m \cup \mathcal{O}_d)) \setminus \{\alpha\} \models \alpha$. As $\alpha \in \mathcal{O}_d$, and because $\mathcal{O} \setminus (\mathcal{O}_m \cup \mathcal{O}_d)$ contains only axioms indispensable in $\mathcal{O}$, we have that $\alpha \in \hat{\mathcal{O}}'$. Moreover, since $\alpha \in \mathcal{O}_d$, we get $\mathcal{O} \setminus \{\alpha\} = (\mathcal{O} \setminus \mathcal{O}_d) \cup (\mathcal{O}_d \setminus \{\alpha\}) \models \alpha$. Since $\mathcal{O}_m$ is a module for $\sigma(\mathcal{O}_d)$ in $\mathcal{O} \setminus \mathcal{O}_d$, the entailment $\mathcal{O}_m \cup (\mathcal{O}_d \setminus \{\alpha\}) \models \alpha$ follows from Definition 2, as $\sigma(\mathcal{O}_d \setminus \{\alpha\}) \cap \sigma(\mathcal{O} \setminus \mathcal{O}_d) \subseteq \sigma(\mathcal{O}_d)$. It implies that $\langle \mathcal{O}_m \cup (\mathcal{O}_d \setminus \{\alpha\}) \rangle = \langle \mathcal{O}_m \cup \mathcal{O}_d \rangle = \langle \hat{\mathcal{O}}' \rangle$, and thus, we get $\hat{\mathcal{O}}' \setminus \{\alpha\} \models \alpha$, since $\alpha \in \langle \mathcal{O}_m \cup (\mathcal{O}_d \setminus \alpha) \rangle$. This contradicts the assumption and $\hat{\mathcal{O}}' \cup (\mathcal{O} \setminus (\mathcal{O}_m \cup \mathcal{O}_d))$ is therefore irredundant. □

According to Proposition 2, computation of reductions of an ontology can be restricted to its dispensable axioms combined with a module in its indispensable axioms computed for the signature of the dispensable axioms. Due to Hypothesis 1 this computation of the module can be expected to potentially filter out large parts of an ontology irrelevant for elimination of redundancy in a pre-computation step when pre-identifying dispensable axioms.

To provide an effective and efficient method for computing all reductions of an ontology, we finally introduce the notion of a *reduction tree* based on principles of the hitting set tree algorithm presented in [12] and applied to computing justifications in [9].

Definition 6 (reduction tree). *For an ontology $\mathcal{O}$ with indispensable axioms $\mathcal{O}_i$, a* reduction tree *is a tree structure T with nodes labelled by sets of axioms and arcs labelled by axioms. Let $\mathcal{O}^* := \mathcal{O}_i \cup \{\alpha \in \mathcal{O} \mid \mathcal{O}_i \not\models \alpha\}$, $P(n)$ be the set of axioms along the path of arcs from the root node of T to node n, $\mathcal{O}_d(n)$ be the axioms dispensable in $\mathcal{O}^* \setminus P(n)$, $\mathcal{O}_d(n, \alpha)$ be the axioms of $\mathcal{O}_d(n)$ dispensable in $\mathcal{O}^* \setminus (P(n) \cup \{\alpha\})$, $\mathcal{O}_m(n)$ be a module for $\sigma(\mathcal{O}_d(n))$ in $\mathcal{O}^* \setminus P(n)$ and $\mathcal{O}_m(n, \alpha)$ be a module for $\sigma(\mathcal{O}_d(n, \alpha))$ in $\mathcal{O}_m(n)$. The tree T is defined recursively:*

- *T has at least one node, the root node n_0, which is labelled by $\hat{\mathcal{O}}' \cup (\mathcal{O}_i \setminus \mathcal{O}_m(n_0))$ for some reduction $\hat{\mathcal{O}}'$ of $\mathcal{O}_m(n_0) \cup \mathcal{O}_d(n_0)$;*
- *if n is a node of T with node label $\hat{\mathcal{O}}_n$ then n has a successor node n_α for each axiom α in $\hat{\mathcal{O}}_n \cap \mathcal{O}_d(n)$ with the following properties:*
  - *n_α is connected to n by an edge labelled by α;*
  - *the label of n_α is $\hat{\mathcal{O}}' \cup (\mathcal{O}_i \setminus \mathcal{O}_m(n, \alpha))$ for some reduction $\hat{\mathcal{O}}'$ of $\mathcal{O}_m(n, \alpha) \cup \mathcal{O}_d(n, \alpha)$.*

We show that a reduction tree can be used as a means to compute all reductions of an ontology in the following theorem.

Theorem 2. *The set of all node labels of a reduction tree for an ontology $\mathcal{O}$ is the set of all reductions of $\mathcal{O}$.*

Proof. We will prove the following two claims: a) the label of any node in T is a reduction of $\mathcal{O}$; b) for any reduction $\hat{\mathcal{O}}$ of $\mathcal{O}$ there is a node in T with label $\hat{\mathcal{O}}$.

a) The proof is by induction over the tree structure of T.

The label $\hat{\mathcal{O}}'\cup(\mathcal{O}_i\setminus\mathcal{O}_m(n_0))$ of n_0 is equivalent to $\hat{\mathcal{O}}'\cup(\mathcal{O}^*\setminus(\mathcal{O}_m(n_0)\cup\mathcal{O}_d(n_0)))$, since $\mathcal{O}^*\setminus\mathcal{O}_d=\mathcal{O}_i$. Due to Proposition 2 $\hat{\mathcal{O}}'\cup(\mathcal{O}^*\setminus(\mathcal{O}_m(n_0)\cup\mathcal{O}_d(n_0)))$ is a reduction of $\mathcal{O}^*$, and due to Lemma 1 it is also a reduction of the semantically equivalent $\mathcal{O}$.

Now, let n be any node in T with label $\hat{\mathcal{O}}_n$ a reduction of $\mathcal{O}$ and n_α be any successor node of n connected to n by an edge labelled α. Then, the node label $\hat{\mathcal{O}}:=\hat{\mathcal{O}}'\cup(\mathcal{O}_i\setminus\mathcal{O}_m(n,\alpha))$ of n_α is equivalent to $\hat{\mathcal{O}}'\cup((\mathcal{O}^*\setminus(P(n)\cup\{\alpha\}))\setminus(\mathcal{O}_m(n,\alpha)\cup\mathcal{O}_d(n,\alpha)))$, since $\mathcal{O}^*\setminus(P(n)\cup\{\alpha\}\cup\mathcal{O}_d(n,\alpha))=\mathcal{O}_i$. Hence, $\hat{\mathcal{O}}$ is a reduction of $\mathcal{O}^*\setminus(P(n)\cup\{\alpha\})$ due to Proposition 2, since $\mathcal{O}_d(n,\alpha)$ are just the dispensable axioms in $\hat{\mathcal{O}}$. Due to Lemma 2 $\hat{\mathcal{O}}$ is also a reduction of $\mathcal{O}^*\setminus P(n)$, because of $\alpha\in\mathcal{O}_d(n)$. Since also $\hat{\mathcal{O}}_n$ is a reduction of $\mathcal{O}^*\setminus P(n)$ (due to Proposition 2), we have that $\langle\hat{\mathcal{O}}_n\rangle=\langle\hat{\mathcal{O}}\rangle$. Hence, the irredundant $\hat{\mathcal{O}}$ is also a reduction of $\mathcal{O}$, as is $\hat{\mathcal{O}}_n$.

b) Let $\hat{\mathcal{O}}$ be a reduction of $\mathcal{O}$. Again by induction, we show that $\hat{\mathcal{O}}$ is a node label along some branch in T.

For the root node n_0 we have that $\hat{\mathcal{O}}\subseteq\mathcal{O}^*=\mathcal{O}^*\setminus P(n_0)$. Now, let n be any node in T with $\hat{\mathcal{O}}\subseteq\mathcal{O}^*\setminus P(n)$. If the label of n is not $\hat{\mathcal{O}}$ then it contains some axiom α with $\alpha\notin\hat{\mathcal{O}}$, since different reductions of $\mathcal{O}$ deviate by conditionally dispensable axioms due to Proposition 1. In this case, there is a successor node n_α, connected to n by an edge labelled α, such that $\hat{\mathcal{O}}\subseteq\mathcal{O}^*\setminus(P(n_\alpha))$.

Due to the strict decrease of the set $\mathcal{O}^*\setminus P(n)$ down the paths the tree is finite and at a certain point $\hat{\mathcal{O}}$ must be the label of some node. $\square$

With Algorithm 5 and Algorithm 4 we present procedures to compute all reductions of an ontology in an optimized way. A call to the procedure computeReductions initiates the computation with a call to the recursive procedure computeAllReductions, which traverses the reduction tree and makes use of Lemma 2 when passing subsets of dispensable axioms down the tree structure. The procedure computeModule computes a module of an ontology based on a signature with the properties according to Definition 2 using techniques from [4], while the procedure determineUncondDispAxioms in Algorithm 6 identifies part of the unconditionally dispensable axioms according to Lemma 1. In addition to pre-identification of dispensable axioms and module extraction, we can also make use of optimizations for node label reuse and tree pruning that have been devised for hitting set tree algorithms. They are described in [12] and are implemented in the conditions of the if-statements in Algorithm 4, similar to [9].

5 Oberservations on Reduction Computation

In order to evaluate the practicability of removing redundancy from ontologies, the above algorithms were implemented using the latest version of the OWL API[3]. We used the Pellet reasoner [14] (v2.1.1) to check entailments, compute regular justifications and to extract modules. The tests have been performed

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

Algorithm 4. computeAllReductions ($\mathcal{O}$, $\mathcal{O}_d$, $\mathcal{O}_m$, P; R, P_c, P_o) – Collect all reductions of an ontology recursively

```
Require: an ontology $\mathcal{O}$, the set $\mathcal{O}_d$ of axioms dispensable in $\mathcal{O}$, a module $\mathcal{O}_m$ for
  $\sigma(\mathcal{O}_d)$ in $\mathcal{O} \setminus \mathcal{O}_d$, a set $P$ of axioms with $P \cap \mathcal{O} = \emptyset$
Ensure: $R$ contains all reductions of $\mathcal{O}$, $P_c \dots$, $P_o \dots$
  if $R \neq \emptyset$ and $P \cap \mathcal{O}^* = \emptyset$ for any $\mathcal{O}^* \in R$ then
    $\hat{\mathcal{O}} := \mathcal{O}^*$
  else
    $\hat{\mathcal{O}} := \emptyset$
    computeSingleReduction ($\mathcal{O}_m \cup \mathcal{O}_d$, $\mathcal{O}_d$; $\hat{\mathcal{O}}$)
    $\hat{\mathcal{O}} := \hat{\mathcal{O}} \cup (\mathcal{O} \setminus \mathcal{O}_m)$
    $R := R \cup \{\hat{\mathcal{O}}\}$
  endif
  $\mathcal{O}'_m := \mathcal{O}'_d := \emptyset$
  for all $\alpha \in \hat{\mathcal{O}} \cap \mathcal{O}_d$ do
    if not ($P \cup \{\alpha\} \supseteq P^*$ for any $P^* \in P_c$ or $P \cup \{\alpha\} = P^*$ for any $P^* \in P_o$) then
      determineDispensableAxioms ($\mathcal{O} \setminus \{\alpha\}$, $\mathcal{O}_d \setminus \{\alpha\}$; $\mathcal{O}'_d$)
      computeModule ($\mathcal{O}_m$, $\sigma(\mathcal{O}'_d)$; $\mathcal{O}'_m$)
      computeAllReductions ($\mathcal{O} \setminus \{\alpha\}$, $\mathcal{O}'_d$,$\mathcal{O}'_m$, $P \cup \{\alpha\}$;$R$, $P_c$, $P_o$)
    endif
    $P_o := P_o \cup \{P \cup \{\alpha\}\}$
  end for
  $P_c := P_c \cup \{P\}$
```

Algorithm 5. computeReductions ($\mathcal{O}$; R) – Calculate all reductions of an ontology

```
Require: an ontology $\mathcal{O}$
Ensure: $R$ contains all reductions of $\mathcal{O}$
  $\mathcal{O}_d := \emptyset$
  determineDispensableAxioms ($\mathcal{O}$, $\mathcal{O}$; $\mathcal{O}_d$)
  $\mathcal{O}_m := \emptyset$
  computeModule ($\mathcal{O} \setminus \mathcal{O}_d$, $\sigma(\mathcal{O}_d)$; $\mathcal{O}_m$)
  $\mathcal{O}_u := \emptyset$
  determineUncondDispAxioms ($\mathcal{O}_m \cup \mathcal{O}_d$, $\mathcal{O}_d$; $\mathcal{O}_u$)
  $R := P_c := P_o := \emptyset$
  computeAllReductions ($\mathcal{O} \setminus \mathcal{O}_u$, $\mathcal{O}_d \setminus \mathcal{O}_u$, $\mathcal{O}_m$, $\emptyset$; $R, P_c, P_o$)
```

Algorithm 6. determineUncondDispAxioms ($\mathcal{O}$, $\mathcal{O}_d$; $\mathcal{O}_u$) – Determine unconditionally dispensable axioms

```
Require: an ontology $\mathcal{O}$, the set $\mathcal{O}_d$ of axioms dispensable in $\mathcal{O}$
Ensure: $\mathcal{O}_u$ contains only axioms that are unconditionally dispensable in $\mathcal{O}$
  $\mathcal{O}_u := \emptyset$
  for all $\alpha \in \mathcal{O}_d$ do
    if $\mathcal{O} \setminus \mathcal{O}_d \models \alpha$ then $\mathcal{O}_u := \mathcal{O}_u \cup \{\alpha\}$
  end for
```

Table 2. Results of redundancy elimination with all optimizations enabled.

				reduced to		dispensability				
Ontology	DL	$\#\mathcal{O}$	$\#\{\hat{\mathcal{O}}_i\}$	max.	min.	$\#\mathcal{O}_i$	$\#\mathcal{O}_c$	$\#\mathcal{O}_u$	t_{first}	t_{mean}
1 HumanRel	$\mathcal{ALCIQ}$	14	2	50%	57%	6	3	5	0.19	0.10
2 Generations	$\mathcal{ALCOIF}$	38	5	87%	87%	25	2	11	0.10	0.04
3 Nautilus	$\mathcal{ALCHF(D)}$	38	1	84%	84%	32	0	6	0.06	0.06
4 PeriodicTable	$\mathcal{ALU}$	58	1	97%	97%	56	0	2	0.29	0.29
5 People	$\mathcal{ALCHOIN}$	108	1	93%	93%	88	0	20	0.52	0.52
6 DOLCE Lite	$\mathcal{SHIF}$	351	58	54%	56%	134	157	60	7.73	4.61
7 Pizza	$\mathcal{SHOIN}$	712	12	58%	59%	404	26	282	20.38	2.21
8 Transportation	$\mathcal{ALCH(D)}$	1157	3	90%	90%	1011	6	140	32.10	10.87
9 Economy	$\mathcal{ALCH(D)}$	1625	3	43%	44%	705	4	916	43.46	21.74
10 Process	$\mathcal{ALCHOF(D)}$	2578	15	94%	94%	2210	29	339	172.66	14.56
11 Wine	$\mathcal{SHOIN(D)}$	657	3	40%	40%	262	5	390	338.99	57.30
12 FlyAnatomy	$\mathcal{EL}^{++}$	10471	5	98%	98%	10289	14	168	2845.76	576.78

on a laptop with 2.4 GHz dual core processor, with Java 1.6, assigning 1 GB memory to Java. A selection of publicly available ontologies (as shown in Table 2) varying in size and expressivity have been used in the experiments.[4]

Before we conducted our main experiments with the optimized hitting set tree approach, we tested the elimination of redundancy via justifications as described in Algorithm 1 on the same set of ontologies. Small and inexpressive ontologies could be reduced properly in acceptable time. Especially, reductions for the human relationship example (1) could be computed in 2,24s, for the Nautilus ontology (3) in 4,98s and for the PeriodicTable (4) in 9,7s. No redundancy was found in the Generations ontology (2) and the People ontology (5) as their redundancies depend on uninternalized RBox axioms. Unfortunatly, computations for all other ontologies in our test set either terminated with heap space exceptions or did not terminate at all within a timeframe of several hours. These results lead us to the belief that a more scalable method both in terms of performance and with no restrictions regarding axiom types is preferable.

As an alternative, we evaluated redundancy elimination with the optimized hitting set tree approach as described in Algorithm 4. As Table 2 shows, the achieved reduction rates ranged from 40% to 98%. In most cases the number of dispensable axioms is considerably smaller than the number of indispensable ones, which empirically supports Hypothesis 1. Exceptions are ontologies designed as example showcases, such as the human relationship or pizza ontology. Further note that Hypothesis 1 was formulated with focus on reducing TBoxes. If we, for example, consider ABoxes with large amounts of materialized knowledge, such as transitive role assertions, the situation might be different and the optimizations would need to be configured accordingly.

The ontologies in Table 2 are sorted according to the time it took for the first reduction to be computed. The quickest work in real-time (1–5), some take up

[4] The wine ontology may be retrieved from `http://www.w3.org/TR/2003/PR-owl-guide-20031209/wine`. All other ontologies used may be found in the TONES ontology repository at `http://owl.cs.manchester.ac.uk/repository`.

to seconds, others take up to minutes, and the last one reveals the computational limits of the approach. The combination of both size and expressivity determine the efficiency of redundancy elimination although the results indicate that the impact of reasoning complexity is considerably higher. Notably, the wine ontology (11) needed longer time to process than ontologies (8–10), which have more axioms but are less expressive; e.g. it took more than seven times longer to compute the first wine (11) reduction than the first economy (9) reduction despite the significantly smaller size. Interestingly the Pizza ontology does not need as long although it is also expressed in $\mathcal{SHOIN}$. It is however known that certain configurations of concept descriptions are especially hard for automated reasoning as it is the case in the wine ontology as discussed by [13].

We investigated the effect of different configurations of optimizations. As expected, the pre-identification of dispensable axioms brings a large performance benefit. The computation time for the first reduction is in general significantly larger than the computation times for successive reductions.

In our current experiments we considered two configurations for module extraction: firstly only an initial module extraction with no further extraction during the hitting set exploration, and secondly the repeated extraction of module throughout the algorithm, and compared these with the computation times when just using hitting-set optimizations. We observed that for small ontologies (1–3) the cost of modularization was higher than the gain. Here the modularization slowered the computation up to factor of two to three. At ontologies (4–9) a break-even seems to be reached for initial modularization while inner modularization is still costly. For the larger and more expressive ontologies we find a gain in modularization, though no clear gain of repeated modularization is visible in the current setting. An introspection of the results showed however that the size of the extracted modules decreased within the first two or three computations and then remained the same or just change very little. For example, in ontology 12 the first three modules have the size 10289, 10133, 9931. As the following extraction steps return modules of constant size 9931 the computation is counterproductive. However, this also indicates a positive effect of repeated modularization for large ontologies but also that a more fine-tuned approach to when to start or stop modularization is desirable.

6 Conclusion

We have introduced notions around redundancy in OWL ontologies and have identified typical cases where redundancy is introduced in the course of ontology evolution in an unanticipated way. We have provided two methods for eliminating redundancy: one utilising readily available techniques of computing justifications and internalization, and another one based on a hitting-set-tree algorithm further optimized by module extraction. We have shown our optimized methods to be effective and efficient on typical ontologies used in the Semantic Web context, based on a prototypical implementation.

For future work, we plan to investigate the elimination of redundancy in parts of axioms to yield a more fine-grained notion of redundancy, similar to work on laconic justifications in [7]. Moreover, we want to target the comparison and evaluation of different reductions and to provide measures that help a knowledge engineer to decide for one.

References

1. Alani, H., Harris, S., O'Neil, B.: Winnowing Ontologies Based on Application Use. In: Sure, Y., Domingue, J. (eds.) ESWC 2006. LNCS, vol. 4011, pp. 185–199. Springer, Heidelberg (2006)
2. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P. (eds.): The Description Logic Handbook: Theory, Implementation and Applications. Cambridge University Press, Cambridge (2007)
3. Bienvenu, M.: Prime Implicate Normal Form for $\mathcal{ALC}$ Concepts. In: AAAI 2008: Proc. of the 23rd Conference on Artif. Intelligence, pp. 412–417. AAAI Press, Menlo Park (2008)
4. Cuenca Grau, B., Horrocks, I., Kazakov, Y., Sattler, U.: Extracting Modules from Ontologies: A Logic-based Approach. In: Stuckenschmidt, H., Parent, C., Spaccapietra, S. (eds.) Modular Ontologies. LNCS, vol. 5445, pp. 159–186. Springer, Heidelberg (2009)
5. Furbach, U., Obermaier, C.: Knowledge Compilation for Description Logics. In: Dershowitz, N., Voronkov, A. (eds.) LPAR 2007. LNCS (LNAI), vol. 4790. Springer, Heidelberg (2007)
6. Gutierrez, C., Hurtado, C., Mendelzon, A.O.: Foundations of Semantic Web Databases. In: PODS 2004: Proceedings of the Twenty-third ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, pp. 95–106. ACM, New York (2004)
7. Horridge, M., Parsia, B., Sattler, U.: Laconic and Precise Justifications in OWL. In: Sheth, A.P., Staab, S., Dean, M., Paolucci, M., Maynard, D., Finin, T., Thirunarayan, K. (eds.) ISWC 2008. LNCS, vol. 5318, pp. 323–338. Springer, Heidelberg (2008)
8. Horrocks, I., Kutz, O., Sattler, U.: The Even More Irresistible $\mathcal{SROIQ}$. In: Proc. of the 10th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR 2006), pp. 57–67. AAAI Press, Menlo Park (2006)
9. 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.) ASWC 2007 and ISWC 2007. LNCS, vol. 4825, pp. 267–280. Springer, Heidelberg (2007)
10. Liberatore, P.: Redundancy in Logic I: CNF Propositional Formulae. Artif. Intell. 163(2), 203–232 (2005)
11. W3C, O.W.L.: Working Group. OWL 2 Web Ontology Language: Document Overview. W3C Recommendation, October 27 (2009), `http://www.w3.org/TR/owl2-overview/`
12. Reiter, R.: A Theory of Diagnosis from first Principles. Artif. Intell. 32(1), 57–95 (1987)
13. Sirin, E., Cuenca Grau, B., Parsia, B.: From Wine to Water: Optimizing Description Logic Reasoning for Nominals. In: KR, pp. 90–99. AAAI Press, Menlo Park (2006)
14. Sirin, E., Parsia, B., Cuenca Grau, B., Kalyanpur, A., Katz, Y.: Pellet: A Practical OWL-DL Reasoner. Journal of Web Semantics 5(2), 51–53 (2007)

Evaluating the Stability and Credibility of Ontology Matching Methods

Xing Niu[1], Haofen Wang[1], Gang Wu[2], Guilin Qi[3], and Yong Yu[1]

[1] Shanghai Jiao Tong University
{xingniu,whfcarter,yyu}@apex.sjtu.edu.cn
[2] Northeastern University
wugang@ise.neu.edu.cn
[3] Southeast University
gqi@seu.edu.cn

Abstract. Ontology matching is one of the key research topics in Semantic Web. In the last few years, many matching methods have been proposed to generate matches between different ontologies either automatically or semi-automatically. To select appropriate ones, users need some measures to judge whether a method can achieve the similar compliance even on one dataset without reference matches and whether such a method is reliable w.r.t. its output result along with the confidence. However, widely-used traditional measures like *precision* and *recall* fail to provide sufficient hints. In this paper, we design two novel evaluation measures to evaluate **stability** of matching methods and one measure to evaluate **credibility** of matching confidence values, which help answer the above two questions. Additionally, we carry out comparisons among several carefully selected methods systematically using our new measures. Besides, we report some interesting findings such as identifying potential defects of our subjects.

1 Introduction

An ontology provides a common vocabulary, which can be used to model a domain. It is a formal representation of shared knowledge between human beings and machines. Nevertheless, due to the openness of Semantic Web, the widely adoption of ontologies also exposes the heterogeneity problem to the public. Ontology matching is one of the positive efforts to reduce heterogeneity. It aims at finding matches between semantically related entities of different ontologies[7]. In recent years, many ontology matching methods and systems have been developed such as Lily[15], ASMOV[11], Anchor-Flood[9] (aflood for short), RiMOM[12], Falcon-AO[10] and AgreementMaker[2] (AgrMaker for short). With so many matching methods in hand, users need some criteria to evaluate them and select appropriate ones for their applications.

Users usually choose match candidates above a *confidence threshold* (CT). The threshold is adjusted on some training datasets with reference matches on hand. Then the trained CT is applied on test data. To achieve good matching

© Springer-Verlag Berlin Heidelberg 2011

performance, data characteristic or distribution of the test data is assumed to be similar to that of the training set. However, it does not always hold. Additionally, it is labor-intensive and error-prone to obtain comprehensive reference matches especially between large ontologies. In some cases, we even do not have authorizations to access the whole ontologies to get those matches. Thus, a widely-used feasible way is to train *CT* on small or partial datasets with representative selected samples. When trying to select the most appropriate method or system according to the user's application domain, we can only predict the matching quality of those methods working on ontologies without any reference matches. Hence, we introduce a new concept called **stability** for matching assessment. Generally speaking, a matching method with high stability indicates that it performs consistently on the data of different domains or scales.

On the other hand, a matching method outputs candidate matches sorted by their matching confidence values. We prefer methods which generate true positive matches with high confidence values (ranked high) while return false positive ones with low values (ranked low). This is another important criteria called **credibility** to guide users on ontology matching method selection.

However, existing measures only focus on judging the basic compliance like precision, recall or their extensions like relaxed precision and recall[4], and semantic precision and recall[6]. In this way, these measures fail to assess a matching method based on the two important criteria, which makes users hard to select the most suitable method or system. Some methods[5,13,14] for selecting the best matching strategy take several parameters into account, yet these parameters do not include scores for evaluating stability and credibility.

In this paper, we propose several novel measures according to the matching criteria. The main contributions are as follows:

– We propose the *STD* (STandard Deviation) score to measure matching stability by examining the fluctuation of the original *confidence threshold*s. To give an overall consideration of both the theoretical optimum matching quality of matching methods and their matching quality using the trained *CT*s in practical, we extend the classical *F-measure* to a comprehensive version.
– A *ROC* (Receiver Operating Characteristics) graph indicates the tradeoff between benefits (true positive matches) and costs (false positive matches) [8]. We use the *ROC-AUC* (Area Under Curve) score to measure the credibility by taking the tradeoff into account.
– We further carry out comprehensive experiments to compare several carefully selected methods based on the new measures mentioned above. By observing exceptions, e.g., the score of a matching method is well below the average, we can tell potential weak points a matching method has on particular datasets.

The rest of this paper is organized as follows. Section 2 presents the formal representation of matches and describes some typical ontology matching methods. In Section 3, we recap some basic measures as well as introduce the new measures. In Section 4, we show experimental results using the new introduced measures and provide deep analysis. Finally, we make a conclusion in Section 5.

2 Preliminaries

We will introduce some typical ontology matching methods in a nutshell. They are outstanding participants of OAEI (Ontology Alignment Evaluation Initiative) campaigns[1] and subjects of our evaluation experiments.

Lily[15] Lily realizes three main matching components: GOM (Generic Ontology Matching) uses semantic subgraph technique to combine lexical and structural matching; LOM (Large-scale Ontology Matching) is adopted to match large size ontologies without using ontology modularization; SOM (Semantic Ontology Matching) uses the Web knowledge to recognize the semantic relations through the search engine.

ASMOV[11] ASMOV incorporates a semantic validation process (remove any incorrect or invalid matches) and calculates four partial similarities: sim_L measures lexical similarity referring WordNet; sim_I and sim_E measure internal and external structure similarity respectively; sim_N measures individual one.

Anchor-Flood[9] Aflood aims at achieving high performance and resolving the scalability problem. This algorithm starts off with a pair of concepts from each ontology, gradually exploring concepts by collecting neighboring concepts. This system has two parts: one is the ontology schema matching algorithm that aligns concepts and properties; the other is the instance matching approach.

RiMOM[12] RiMOM includes several lexical matching strategies: Edit-Distance, Vector-Distance and referring to WordNet. The structural matching uses an adaptive variation of the similarity flooding. A strategy selection process is applied to improve the match accuracy. Similarity combination and propagation are also key steps.

Falcon-AO[10] Falcon-AO consists of four matchers: I-Sub and V-Doc (Virtual Documents) are two light-weight linguistic matchers which take neighboring information into consideration; GMO (Graph Matching for Ontologies) is a structural matcher, uses RDF bipartite graphs to represent ontologies and computes structural similarities; PBM (Partition-Based Block Matching) divides large-scale ontologies into blocks and finds block matches between them.

AgreementMaker[2] AgrMaker adapts three string-based techniques: BSM (Base Similarity Matcher), PSM (Parametric String-based Matcher) and VMM (Vector-based Multi-word Matcher), VMM is similar to V-Doc. A graph-based structural matcher is used which is based on the idea that if two nodes are matched with a high similarity, then their children should be similar. It also refers to WordNet to find further matches as its last step.

3 Evaluation Measures

In this section, we propose some new measures for evaluating matching methods. We first introduce three well-known measures: *Precision*, *Recall* and *F-measure*.

[1] http://oaei.ontologymatching.org/

Let the result set of ontology matching be divided into subset *TP* (true positives, means correctly proposed matches) and subset *FP* (false positives, means falsely proposed matches). Let *FN* (false negatives) be the set of missing correct matches, then it comes the definitions of *Precision* and *Recall*:

$$Precision = |TP|/(|TP| + |FP|) \tag{1}$$

$$Recall = |TP|/(|TP| + |FN|) \tag{2}$$

Precision reflects the share of correct matches among all found ones, while *Recall* reflects the share of correctly proposed matches among all expected ones. In order to avoid the imperfection in evaluating the compliance with *Precision* or *Recall* alone, *F-measure*, the harmonic mean of *Precision* and *Recall* is proposed and widely used:

$$F\text{-}measure = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{3}$$

3.1 Comprehensive *F-measure*

A confidence value is included in the match computed from matching methods to describe the confidence about the result. The higher a confidence value is, the more similarities two entities share, and the more correct a match is likely to be. Intuitively speaking, matches with low confidence value are likely to be incorrect and will lower the *Precision* values. However, denying all matches with low confidence values to increase *Precision* values will lose a certain number of correct ones, which, on the contrary, will decrease the *Recall* values.

Therefore, in practice, application systems have to determine a *confidence threshold* (*CT*) value to preserve only those matches with confidence values above it in a matching task. In this case, *F-measure* is a function of *CT*. The greedy principle of selecting the *CT* is to have *F-measure* reaches its maximum [7]:

$$maxF\text{-}measure = \max F\text{-}measure(CT) \tag{4}$$

$$maxFCT = \arg\max_{CT} F\text{-}measure(CT) \tag{5}$$

The *maxF-measure* reflects the theoretical optimal matching quality of matching methods but it does not consider their matching quality in practice. The design of a matching method usually takes several semantic information into account, such as Lexical knowledge, Structural knowledge, Domain knowledge, and Instance-based knowledge [1], which are quite dataset specifically (as shown in our subsequent experiments). Thus, application systems have to adjust and tune the *CT* dramatically to keep a matching method applicable with a high *F-measure* across different datasets. This is costly especially facing large-scale real-world datasets, and even infeasible if there are no enough reference answers for certain datasets in hand. In this case, those matching methods that generate relative stable *maxFCT*s will surpass the others, because they can ensure to give a holistic applicable $maxFCT$ value from only some selected ones among all datasets. Since validation datasets are scarce, we propose a novel measure, i.e. *uniF-measure* (*uniform F-measure*), to simulate the practical application and evaluate such stability of matching methods.

The *uniF-measure* is computed as follows. First, we compute $maxFCT$s from a set of selected datasets (named a **test unit**), and then calculate their arithmetic mean average($maxFCT$s). Further, the *F-measure* for each corresponding dataset is re-computed under this mean value, which is then defined as the *uniF-measure*:

$$uniF\text{-}measure = F\text{-}measure(\text{average}(maxFCT\text{s})) \tag{6}$$

The test unit mentioned above usually means a set of similar datasets describing the same domain and sharing many resemblances, but differing on details. A test unit contains several matching tasks and reference answers are necessary. It is hired to simulate featured subsets of a particular ontology.

If a matching method is stable, $maxFCT$s of a test unit should be relatively stable, i.e. *uniF-measure* value should not be much lower than *maxF-measure* value. In this way, we can observe the stability of matching methods by contrasting their *maxF-measure* values and *uniF-measure* values, or just calculating their arithmetic mean, the *comF-measure* (*comprehensive F-measure*) value. *comF-measure* is an extended *F-measure* and defined by

$$comF\text{-}measure = (maxF\text{-}measure + uniF\text{-}measure)/2. \tag{7}$$

3.2 *STD* Score

As described in Section 3.1, for some reason (such as lacking enough reference answers, or in a predicting task), there is only a limited part of the datasets (so-called the test unit) available beforehand. So we select a matching method with high stability on a limited dataset and hope this method can perform well in the coming matching tasks.

Measuring the dispersion of $maxFCT$s in a test unit can help us estimate the difficulty in obtaining *maxF-measure* for a given matching method. However, sometimes it is too strict to just focus on $maxFCT$s. In practice, we also accept those CT values that cause the test unit to have *F-measure* values close to *maxF-measure*. Here we use a real example as shown in Figure 1 to further explain this condition. When $CT = 0.848$, *F-measure* reaches its maximum 0.946. Actually, some smaller values near this *maxF-measure* are fully acceptable too, e.g., $F\text{-}measure = 0.939$ when $CT = 0.878$.

Thus, we give some grace when finally determine the expecting CT measuring. From all CTs that can get top 20 percents of *F-measure*, we choose the maximum CT as the $relaxedCT$. The $relaxedCT$ can be formally defined similar to $maxFCT$:

$$relaxedCT = \max(\arg\underset{CT}{\text{top20pct}}\,(F\text{-}measure(CT))) \tag{8}$$

The dotted box in Figure 1 encloses top 20 percents of *F-measure* and we have $relaxedCT = 0.878$ now. The value of $relaxedCT$ will be too far apart

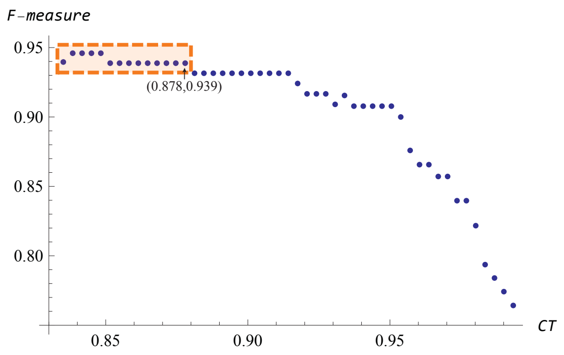


Fig. 1. Relation between *F-measure* and *CT* (result for Benchmark#304 by Falcon-AO)

from *maxFCT* if the range of *F-measure* varies widely in some cases. So we should restrict the range of *relaxedCT* under this condition. Here we choose the top 10 percents of *F-measure* if the variation range exceeds 0.1. If the variation range still exceeds 0.1 under the circumstances, only those *F-measure*s that are between *maxF-measure* and *maxF-measure*−0.1 will be taken into account.

With the definition of *relaxedCT*, we propose the *STD score* as another stability measure.

$$STD\,score = 1 - \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(relaxedCT_i - \overline{relaxedCT}\right)^2} \tag{9}$$

where N is the number of matching tasks in a test unit, and $\overline{relaxedCT}$ is the mean value of *relaxedCT*s.

The *STD* score is a standard deviation to measure the dispersion of *relaxedCT*s, and hence that of *maxFCT*s. According to the probability theory and statistics, slight diversity of *relaxedCT* will cause the deviation of *STD* score apparently. Therefore, *STD* score is effective in measuring the stability of matching methods. *STD* scores are comparable when they are calculated under the same test unit.

3.3 *ROC-AUC* Score

A reasonable matching method should have a certain degree of credibility in confidence value, because the credible confidence value plays an important role in ranking matching results reasonably. With such credibility, the greater confidence value a matching method gives, the relevant match is more likely to be correct. We hire the Receiver Operating Characteristics (*ROC*) analysis technique [8] to measure such credibility.

The *ROC* analysis is used for ‘visualizing, organizing and selecting classifiers based on their performance’ [8]. As shown in Figure 2, *ROC* graphs are two-dimensional polygonal line graphs in which true positives rate is plotted on the Y axis, and false positives rate is plotted on the X axis.

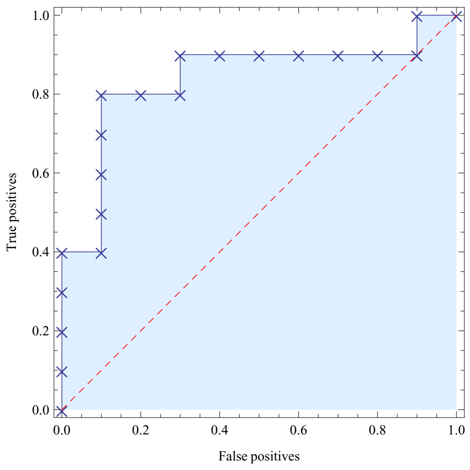


Fig. 2. An Example of *ROC* Graph

To measure the credibility of confidence value with *ROC* graph, we should first sort matches by their confidence values before drawing the *ROC* curves. Starting from the match with the highest confidence value, if it is true positive, the curve climbs up by one unit distance ($1/|TP|$); otherwise, the curve move horizontally to the right by $1/|FP|$. The whole polygonal line starts from origin and ends up at (1,1). The *ROC-AUC* (Area Under Curve) is the area surrounded by *ROC* curve, X axis and line $X = 1$. Since the domains of axes are normalized, the score of this kind of measure could be defined easily as:

$$ROC\text{-}AUC\, score = ROC\text{-}AUC \tag{10}$$

ROC-AUC score is applicable for measuring the credibility of confidence value because if confidence values are really credible, the *ROC* curve should climb straight towards (0,1) in the early stage of drawing the curve, and *ROC-AUC* should be relatively large, i.e. *ROC-AUC* score should be high. The meaning of *ROC-AUC* score could be explained by referring to an illustration from the realm of information retrieval that the users always want to receive most correct answers with high confidence value (cf. ranking score).

Ehrig [3] does not recommend this measure for the reason that different dataset results cannot be compared directly because the unit distance we use for drawing the curve is dependent on the result size. In fact, we do not directly compare matching methods using this measure, but qualitatively analyze their potential problems.

4 Experimental Results

OAEI is widely used for ontology matching evaluation. It not only assesses the strength and weakness of matching methods but also helps improving the work on ontology matching. We choose OAEI 2009 Benchmark track and the Conference track as test datasets because relatively complete reference matches of these tracks are available for judging. In the following subsections, we will introduce main characteristics of each dataset and present experimental results using the above described evaluation measures.

4.1 Testing on the Benchmark Track

The OAEI setups benchmark test library to offer a set of tasks which are 'wide in feature coverage, progressive and stable'[2]. These tasks describe bibliographic reference.

Most ontologies of the Benchmark track orient from a complete reference ontology and there are six categories of alteration. All the benchmark tasks can be divided into five groups and we treat each group as a test unit (tasks in a test unit share many resemblances):

#101-#104 (10X for short) The descriptions of classes or properties may have some differences.

#201-#210 (20X for short) Local names and comments information are reduced while structural information remains the same.

#221-#247 (22X for short) Hierarchy, instances, properties and classes information are reduced while lexical information remains the same.

#248-#266 (24X for short) Lexical information is suppressed and structural information is reduced. We abandon this test unit because we do not think information of real-world ontologies is so deficient.

#301-#304 (30X for short) Four real-world ontologies.

The synthetic benchmark is instrumental in evaluating matching methods. Here we measure a simple matching method and five sophisticated ones. They are StringDistance, Falcon-AO, Lily, ASMOV, RiMOM and aflood respectively. More precisely, StringDistance simply compares differences between strings. We treat it as our baseline. The five sophisticated matching methods are selected due to their outstanding and stable performance in Benchmark track of OAEI campaigns in recent years (2005-2009).

10X Test Unit. This test unit is relative simple, so we provide all three kinds of scores in Table 1. In order to visually compare these scores of different matching methods, we also present results with Spider Charts (Figure 3). In Spider Charts, we can not only capture disparities of scores shown in three axes, but also learn overall quality by surveying the area surrounded by score lines. We start axes of Spider Charts at 0.5 for enlarging the disparities.

The Spider Chart for 10X test unit (Figure 3a) shows that all matching methods get high scores in *comF-measure* and *ROC-AUC* measures except StringDistance. However, *STD* scores of Lily and ASMOV are lower than that of others.

[2] http://oaei.ontologymatching.org/2009/benchmarks/

Table 1. Results for Benchmark Test Units

Tools	#	*comF-measure*	*STD* score	*ROC-AUC*	#	*comF-measure*	*STD* score	*ROC-AUC*
Falcon-AO	10X	0.996537	0.999240	1.000000	22X	0.989778	0.949460	0.996392
Lily		0.993873	0.837004	1.000000		0.966045	0.767347	0.986057
ASMOV		0.991161	0.832834	1.000000		0.795300	0.742211	0.988999
RiMOM		0.965817	0.996060	1.000000		0.986734	0.947575	0.988127
aflood		1.000000	1.000000	1.000000		0.991277	1.000000	0.986153
StringDistance		0.779116	1.000000	0.638158		0.818852	1.000000	0.698579
Falcon-AO	20X	0.897292	0.633577	0.955449	30X	0.806695	0.950244	0.885652
Lily		0.973254	0.914770	0.993886		0.807346	0.836017	0.825668
ASMOV		0.940700	0.836917	0.989130		0.747131	0.762499	0.794369
RiMOM		0.822705	0.472834	0.993197		0.813504	0.892603	0.804618
aflood		0.925260	1.000000	0.973064		0.830853	1.000000	0.900738
StringDistance		0.544872	0.878835	0.439786		0.802029	0.912243	0.548727

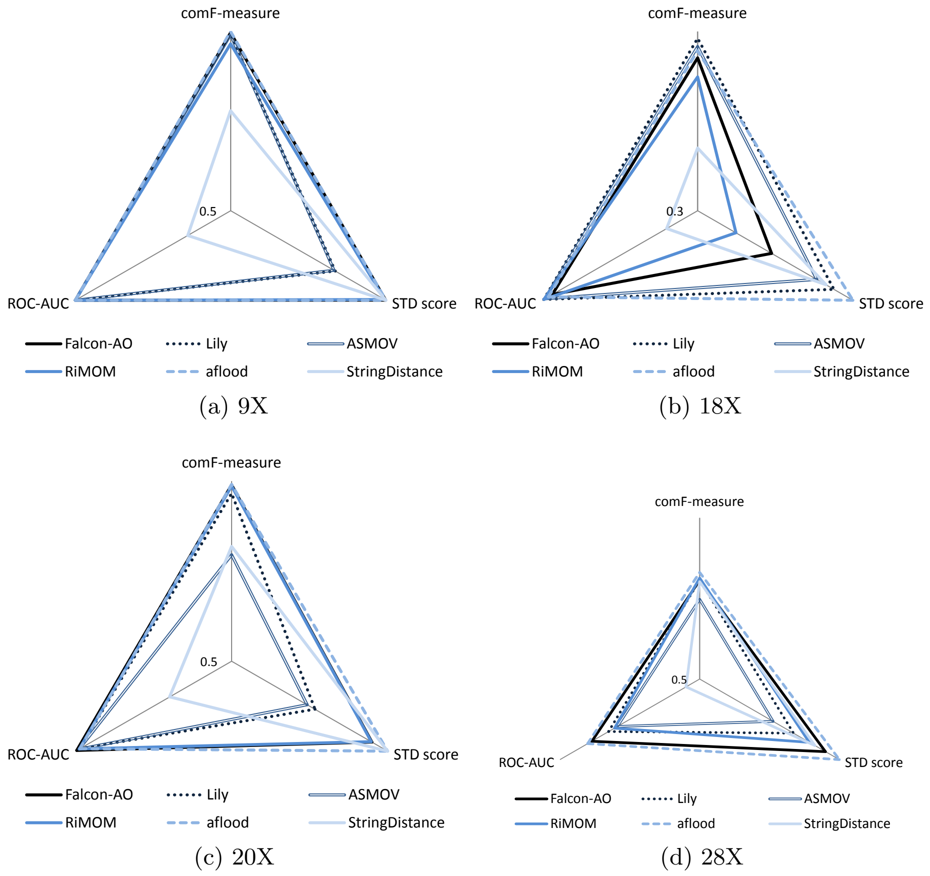
Table 2. Ontology Information of Benchmark-20X Test Unit

Datasets	Names	Comments	Datasets	Names	Comments
203	0	N	207	F	0
204	C	0	208	C	N
205	S	0	209	S	N
206	F	F			

We draw Figure 4 which reflects the fluctuations of *relaxedCT* values, so as for further analysis on the *STD* scores. More precisely, we project each task into one unique point at *X* axis: 10X test unit (including 3 tasks) corresponds to point 1 to point 3, 20X test unit corresponds to 4 to 10, etc. Each horizontal dashed line in red running through a set of points represents the mean value of the corresponding *relaxedCT*s. Intuitively, the more discrete points are, the lower *STD* score will be. We do not present aflood on the figure because *relaxedCT*s of aflood are always 1s. In fact, aflood does not provide distinguishing confidence values but 1 for any match.

20X Test Unit. 20X test unit is a little more complex, and detailed ontology information is described in Table 2. Here we explain the meaning of abbreviations: 0 (stay unchanged), N (suppressed), F (strings in another language than English), C (naming conventions) and S (synonyms). We abandon tasks 201, 202 and 210 due to the same reason of abandoning 24X test unit.

We present *maxF-measure* and *uniF-measure* results (Figure 5) for this test unit because some exceptions are detected. Performances of Falcon-AO (Figure 5a), ASMOV (Figure 5c) and RiMOM (Figure 5d) draw our attention. All of them have clear disparities between *uniF-measure* scores and *maxF-measure* scores in #209. To find out the reason, we can come back to figure 4 (#209 corresponds to 10 at *X* axis). These three matching methods should set *CT*s really low to get *maxF-measure* scores. That is to say, when meet synonyms, these methods do not have much confidence in matching. Comparing with StringDistance we



(a) 9X (b) 18X

(c) 20X (d) 28X

Fig. 3. Spider Charts for Benchmark Test Units

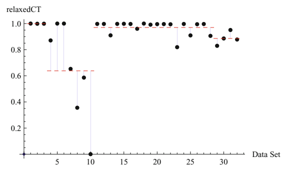
can infer that they must use structural information to get decent *maxF-measure* scores. Lily and aflood are not facing the instability problem although they also meet the challenge of lacking for lexical information.

We also observe that RiMOM does not work well in #205, #206 and #207 at the same time. Together with #209, we can find that RiMOM does not perform stable enough when Local Name information is suppressed a lot.

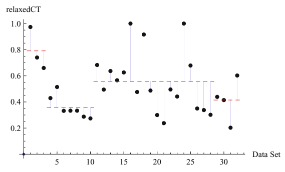
All evaluation results of this test unit are presented in Figure 3b. The overall matching quality can be summarized easily according to comparing areas surrounded by score lines.

22X Test Unit. The detailed ontology information of 22X test unit is described in Table 3. Meanings of abbreviations are: 0 (stay unchanged), N (suppressed), F (flattened), E (expanded) and R (random strings).

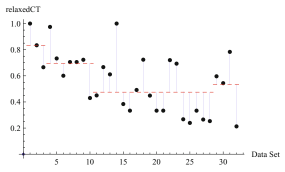
In this test unit, Lily and ASMOV get some exceptions and their *maxF-measure* and *uniF-measure* scores are presented in Figure 6.



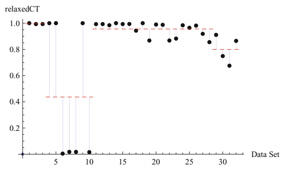
(a) Falcon-AO



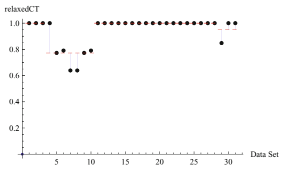
(b) Lily



(c) ASMOV



(d) RiMOM



(e) StringDistance

Fig. 4. The Fluctuations of *relaxedCT* in Benchmark Test Units

Table 3. Ontology Information of Benchmark-22X Test Unit

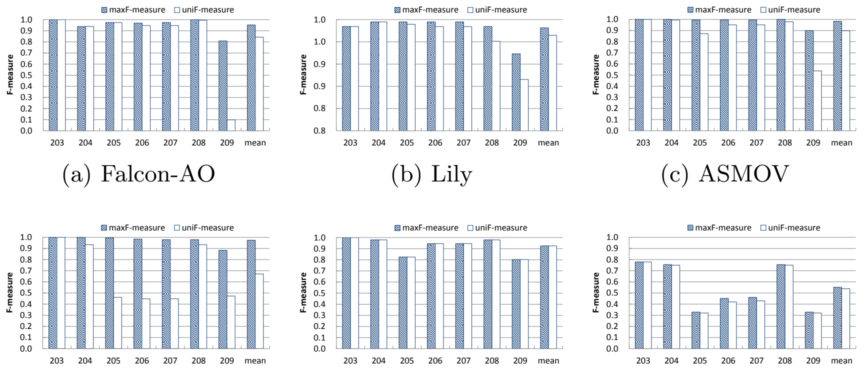
Datasets	Hierarchy	Instances	Properties	Classes	Datasets	Hierarchy	Instances	Properties	Classes
221	N	0	0	0	233	N	0	N	0
222	F	0	0	0	236	0	N	N	0
223	E	0	0	0	237	F	N	0	0
224	0	N	0	0	238	E	N	0	0
225	0	0	R	0	239	F	0	N	0
228	0	0	N	0	240	E	0	N	0
230	0	0	0	F	241	N	N	N	0
231	0	0	0	E	246	F	N	N	0
232	N	N	0	0	247	E	N	N	0

Referring to Figure 4b, Figure 4c and ontology features described in Table 3, we can find out potential flaws of these matching methods: Lily is hyper-sensitive to instances information while ASMOV is hyper-sensitive to properties information. Hyper-sensitivity means a matching method makes great different in final results (confidence values of matches) for whether or not a certain kind of information is missing. Hyper-sensitivity concededly damages stability of matching methods.

30X Test Unit. This test unit contains four real-world ontologies. According to the explanation of OAEI, the reference matches for these tasks are not perfect[3], so we will not care matching quality which is represented by *comF-measure*. Actually, *maxF-measure* scores of all matching methods, including StringDistance, are very close.

StringDistance does not provide result for #303 because this ontology lacks in local names of classes and properties and StringDistance fails to extract this information from complete URIs. Lily also faces this problem, but it considers

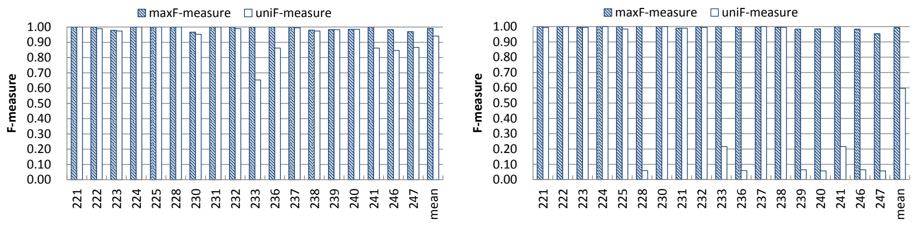
[3] http://oaei.ontologymatching.org/2009/benchmarks/#266



(a) Falcon-AO (b) Lily (c) ASMOV

(d) RiMOM (e) aflood (f) StringDistance

Fig. 5. *maxF-measure* and *uniF-measure* Results for Benchmark-20X Test Unit



(a) Lily (b) ASMOV

Fig. 6. Partial *maxF-measure* and *uniF-measure* Results for B-22X Test Unit

comments, so matches are found after all. Real-world ontologies usually have some special features that matching methods should take into account. Besides missing local names mentioned above, #301 also has a unique characteristic that all local names of properties start with 'has', which is a typical naming convention.

The Spider Char of 30X test unit (Figure 3d) shows that the sophisticated matching methods have no advantages over the simple StringDistance method except when we measure *ROC-AUC* scores.

4.2 Testing on the Conference Track

The ontologies of Conference track come from conference organization domain. All of these ontologies are built by different groups and have 'heterogeneous character of origin'[4]. We choose seven ontologies whose reference matches are offered. The target of this track is to match every two ontologies so there are $C_7^2 = 21$ tasks and we group all these tasks into a single test unit. A simple matching

[4] `http://oaei.ontologymatching.org/2009/conference/`

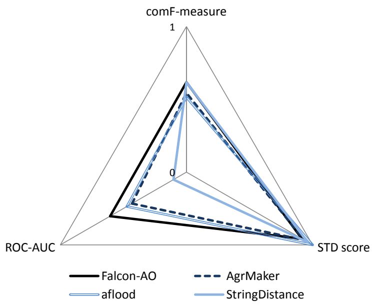


Fig. 7. Spider Chart for Conference Test Unit

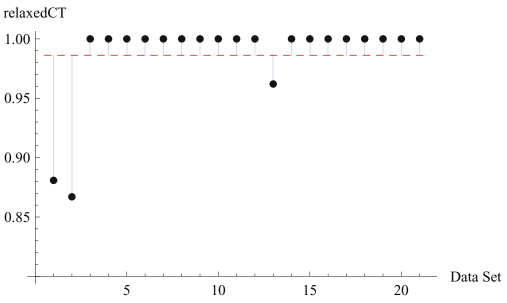


Fig. 8. The Fluctuation of *relaxedCT* in Conference Test Unit (StringDistance)

method (StringDistance) as baseline and three sophisticated ones (Falcon-AO, AgrMaker and aflood) are measured for Conference test unit. Falcon-AO performs well and stably in Conference track of OAEI 2006&2007 while AgrMaker and aflood perform better than other participants in Conference track of OAEI 2008&2009.

From the visual representation of evaluation results (Figure 7), we observe that all *STD* scores are satisfactory while all *comF-measure* scores are not that good. We even notice that simple StringDistance can nearly take the place of the sophisticated ones. However, *ROC-AUC* score holds it back. Figure 8 exhibits that most *relaxedCT*s of StringDistance are 1s, in other words, all confidence values of matches are 1s. That means it is useless to sort matches by confidence values, so users can hardly distinguish more likely correct matches among them.

Aflood faces the same problem since it always sets confidence values as 1s. This method gets normal *ROC-AUC* scores in Benchmark track so we infer that it generates confidence values for sorting during processing but does not output them in the end. Anyhow, it confuses users eventually and does not continue getting high *ROC-AUC* score in this track.

4.3 Discussion

Single *F-measure* score or even *comF-measure* score may mask potential problems of matching methods. As the detailed experimental analysis shows above, differences are manifested on *STD* scores in Benchmark track and *ROC-AUC* scores in Conference track.

Hyper-sensitivity is detected in our experiments by *STD* scores, e.g., Falcon-AO and ASMOV are hyper-sensitive to synonyms, RiMOM is hyper-sensitive to local names and properties information and Lily is hyper-sensitive to instances information. To promote stability of matching methods, researchers should pay more attention to the information that matching methods are hyper-sensitive to.

Users have the authorities to determine final confidence threshold, so matching methods ought to provide credible and diverse confidence values for different matching tasks. Indifference confidence value is a simple cause of low *ROC-AUC* scores, yet more internal factors of matching methods remain to be dug and improved.

5 Conclusions and Future Work

Ontology Matching is one of the most popular research fields in Semantic Web. In recent years, many matching methods have been proposed. In order to assess the matching performance of these methods, it is essential to have a comprehensive benchmark which can find the potential weakness of these methods and help researchers to improve them to a certain extent.

In this paper, we design three new evaluation measures to evaluate stability of matching methods and credibility of their matching confidence values. Moreover, we identify potential defects of subjects by detecting the exception of these measure scores. The deep analysis can shed light on the selection of appropriate matching systems against the specific domain and environment. It may also help pointing out the way to improve a given matching method.

In the future, we intend to extend our evaluation measures to a comprehensive strategy for selecting matching methods and compare this strategy with existing ones. We also plan to test more matching methods under other datasets in OAEI using our proposed measures. As a result, we would like to make both stability and credibility as standard evaluation measures for ontology matching.

Acknowledgments

This work is supported by the National Natural Science Foundation of China under Grant No. 60903010, the Natural Science Foundation of Jiangsu Province under Grant No. BK2009268, and the Key Laboratory of Advanced Information Science and Network Technology of Beijing under Grant No. XDXX1011.

References

1. Bouquet, P., Serafini, L., Zanobini, S.: Semantic coordination: A new approach and an application. In: Fensel, D., Sycara, K., Mylopoulos, J. (eds.) ISWC 2003. LNCS, vol. 2870, pp. 130–145. Springer, Heidelberg (2003)
2. Cruz, I.F., Antonelli, F.P., Stroe, C., Keles, U.C., Maduko, A.: Using agreementmaker to align ontologies for oaei 2009: Overview, results, and outlook. In: Proceedings of the 4th International Workshop on Ontology Matching (2009)
3. Ehrig, M.: Ontology Alignment: Bridging the Semantic Gap, Semantic Web And Beyond Computing for Human Experience, vol. 4. Springer, Heidelberg (2007)
4. Ehrig, M., Euzenat, J.: Relaxed precision and recall for ontology matching. In: Proceedings of the K-CAP 2005 Workshop on Integrating Ontologies (2005)
5. Ehrig, M., Staab, S., Sure, Y.: Bootstrapping ontology alignment methods with APFEL. In: Gil, Y., Motta, E., Benjamins, V.R., Musen, M.A. (eds.) ISWC 2005. LNCS, vol. 3729, pp. 186–200. Springer, Heidelberg (2005)

6. Euzenat, J.: Semantic precision and recall for ontology alignment evaluation. In: Proceedings of the 20th International Joint Conference on Artificial Intelligence, pp. 348–353 (2007)
7. Euzenat, J., Shvaiko, P.: Ontology Matching. Springer, Heidelberg (2007)
8. Fawcett, T.: An introduction to roc analysis. Pattern Recognition Letters 27(8), 861–874 (2006)
9. Hanif, M.S., Aono, M.: An efficient and scalable algorithm for segmented alignment of ontologies of arbitrary size. J. Web Sem. 7(4), 344–356 (2009)
10. Hu, W., Qu, Y.: Falcon-ao: A practical ontology matching system. Web Semantics: Science, Services and Agents on the World Wide Web 6(3), 237–239 (2008)
11. Jean-Mary, Y., Shironoshita, E., Kabuka, M.: Ontology alignment with semantic validation. Web Semantics: Science, Services and Agents on the World Wide Web 7(3), 235–251 (2009)
12. Li, J., Tang, J., Li, Y., Luo, Q.: Rimom: A dynamic multistrategy ontology alignment framework. IEEE Trans. Knowl. Data Eng. 21(8), 1218–1232 (2009)
13. Mochol, M., Jentzsch, A., Euzenat, J.: Applying an analytic method for matching approach selection. In: Proceedings of the 1st International Workshop on Ontology Matching (2006)
14. Tan, H., Lambrix, P.: A method for recommending ontology alignment strategies. In: Proceedings of the 6th International Semantic Web Conference and 2nd Asian Semantic Web Conference, pp. 494–507 (2007)
15. Wang, P., Xu, B.: Lily: Ontology alignment results for oaei 2009. In: Proceedings of the 4th International Workshop on Ontology Matching (2009)

How Matchable Are Four Thousand Ontologies on the Semantic Web

Wei Hu, Jianfeng Chen, Hang Zhang, and Yuzhong Qu

State Key Laboratory for Novel Software Technology, Nanjing University, China
{whu,yzqu}@nju.edu.cn, jf_chen@ymail.com,
hzhang.nju@gmail.com

Abstract. A growing number of ontologies have been published on the Semantic Web by various parties, to be shared for describing things. Because of the decentralized nature of the Web, there often exist different but similar ontologies from overlapped domains, or even within the same domain. In this paper, we collect more than four thousand ontologies and perform a large-scale pairwise matching based on an ontology matching tool. We create about three million mappings between the terms (classes and properties) in these ontologies, and construct a complex term mapping graph with terms as nodes and mappings as edges. We analyze the macroscopic properties of the term mapping graph as well as the derived ontology mapping graph, which characterize the global ontology matchability in several aspects, including the degree distribution, connectivity and reachability. We further establish a pay-level-domain mapping graph to understand the common interests between different ontology publishers. Additionally, we publish the generated mappings online based on the R2R mapping framework. These mappings and our observations are believed to be useful for the Linked Data community in ontology creation, integration and maintenance.

1 Introduction

The *Semantic Web* is an ongoing effort by the W3C Semantic Web Activity for realizing data integration and sharing across different applications and parties. As of today, a growing number of popular *ontologies* have emerged to describe things for specific domains, e.g., the Friend of a Friend (FOAF). These ontologies recommend common classes and properties (uniformly called *terms* in this paper) that are widely and consistently used in data sources.

Because of the decentralized nature of the Web, there usually exist multiple ontologies from overlapped application domains or even within the same domain. In order to establish interoperability between (Semantic) Web applications that use different but related ontologies, *ontology matching* (OM) has been proposed as an effective way for handling the semantic heterogeneity problem. It is useful for many tasks, such as data integration and distributed query processing.

To date, a large amount of (semi-)automatic OM approaches have been proposed in literature [10], which exploit a wide range of characteristics in ontologies,

© Springer-Verlag Berlin Heidelberg 2011

such as linguistic descriptions, structures, data instances, and even background knowledge from thesaurus or third parties' ontologies. But, the global analysis on ontology matchability is still missing, that is, *how matchable are the ontologies on the Semantic Web so far?* In this paper, we dedicate to answering this question. We believe that the study on the morphology of ontology overlaps is important for the Semantic Web, and our observations would help ontology developers and users in the process of ontology creation, integration and maintenance.

Complex network analysis has been widely performed on the page link graph to investigate some macroscopic properties of the Hypertext Web [1,2,5,9]. Recently, such analysis techniques have been applied to the Semantic Web as well, from small sets of ontologies [12,14,18] to the large Linked Data cloud [8,11,17]. However, to the best of our knowledge, because of the high computational cost, the macroscopic matchability among ontologies on the whole Semantic Web has not been well studied yet.

In this paper, we collect more than four thousand Web ontologies by a Semantic Web search engine named Falcons [6], and employ six computers running nearly a year to perform a large-scale pairwise matching by an ontology matching tool named Falcon-AO [16]. We create about 3.1 million mappings between two million terms from the ontologies, and build a complex *term mapping graph*, where nodes are derived from terms and edges are from mappings.

Then, we analyze the macroscopic properties of the term mapping graph in many aspects, including the degree distribution, average distance and clustering coefficient. In addition, we derive an *ontology mapping graph*, in which directed edges are derived from the term mappings with respect to the size of ontologies, for analyzing how big the overlaps of these ontologies are. According to our experiment, we observe that, both the term mapping graph and ontology mapping graph exhibit the scale-free nature with a few "hubs", and the terms (ontologies) from a large part of the graphs form a small world.

Furthermore, we categorize the ontologies in terms of the pay-level-domains of their namespaces (a *pay-level-domain mapping graph*), and observe the common interests among various ontology publishers. We see that `dbpedia.org` and `umbc.edu` are two generic ontology publishers and their ontologies cover a broad range of real-world domains. In addition, our created mappings are all published online[1] based on the R2R mapping specification [4], which would facilitate the Linked Data community to create, integrate and maintain ontologies. Moreover, we are trying to apply these mappings to enhance our Semantic Web browser[2] by recommending matchable terms to general users.

The rest of this paper is organized as follows. Sect. 2 discusses related work. The dataset, metrics and tools used in the experiment are introduced in Sect. 3. In Sect. 4 and Sect. 5, we analyze the macroscopic properties of the term mapping graph and the derived ontology mapping graph, respectively. We further at a higher level investigate the pay-level-domain mapping graph in Sect. 6. Finally, Sect. 7 summarizes our findings in this paper and points out future work.

[1] `http://ws.nju.edu.cn/mappings/`

[2] `http://ws.nju.edu.cn/explorer/`

2 Related Work

Graph analysis has been extensively studied on page link graphs for the Hypertext Web. Albert, et al. [2] analyzed the distributions of incoming and outgoing links between HTML documents on the Web, and observed the power law tails. Adamic and Huberman [1] observed the small world phenomenon in the largest strongly connected component of the website graph. Broder, et al. [5] confirmed the power law distributions of in-degrees and out-degrees, and discovered that a power law also appears in the distribution of the sizes of connected components. They figured out a “bow-tie” structure as the macroscopic structure of the Web. Even recently, researchers were still studying various datasets to investigate the topological properties of the Web [9].

Graph analysis techniques have been conducted to a single ontology or a set of ontologies. Hoser, et al. [15] illustrated some benefits of applying social network analysis to the SWRC and SUMO ontologies, and discussed how different notions of centrality (e.g., degree, betweenness, eigenvector) describe the core content and structure of an ontology. Theoharis, et al. [18] observed the graph features of 250 ontologies, and claimed that a majority of ontologies with a significant number of properties approximate power laws for the total-degrees, and each ontology has a few focal classes with numerous properties and subclasses. At a larger scale, Gil, et al. [13] combined the ontologies from the DAML ontology library into a single RDF graph, which includes 56,592 nodes and 131,130 edges. They found that this graph is a small world and the cumulative degree distribution follows a power law. Tummarello, et al. [21] observed that the distribution (reuse) of URIs over documents follows a power law. Ding, et al. [8] gave a quantitative analysis of `owl:sameAs` deployment status and used these statistics to focus discussion around its usage in Linked Data.

To the best of our knowledge, there are two works that address the analysis of ontology matchability. Ghazvinian, et al. [12] investigated the morphology of ontology mappings among 207 biomedical ontologies, where the mappings were extracted by similar names. Nikolov and Motta [17] created term mappings from declared coreference association (e.g., `owl:sameAs`) and co-typing, and analyzed a snapshot of the Billion Triple Challenge 2009 containing several hundreds of ontologies. Their mappings hold not only the equivalence relations but also the subsumption, e.g., `movie:actor` co-types with `foaf:Person`. In this paper, we analyze over four thousand ontologies, which is much larger than the sizes of the two previous works. Additionally, we use a general ontology matching approach which does not tailor itself to some specific domains. Our experimental results also show some different observations as compared with [12,17].

3 Experiment Setting

The goal of our work is to investigate the macroscopic matchability of ontologies in a dataset that contains a significant number of mappings. We therefore introduce the notion of ontologies, metrics in the experiment and the ontology matching tool Falcon-AO.

3.1 Statistical Data of Ontologies

An ontology $\mathcal{O}$ is viewed as a triple $\langle id, \mathcal{V}, \mathcal{G} \rangle$, where id is a unique identifier; $\mathcal{V}$ is a vocabulary that consists of a non-empty set of terms (classes and properties) holding a common URI namespace [3]; and $\mathcal{G}$ is an RDF graph that describes the terms in $\mathcal{V}$.

We recognize all the vocabularies and their involved terms from Falcons, and dereference their URIs to obtain the dereferenced documents. The identifier of an ontology is the URI namespace of its vocabulary, and the RDF triples in all dereferenced documents are merged as the RDF graph of that ontology, because the dereferenced documents for a vocabulary and its involved terms could be different. Terms having the same namespace as $\mathcal{O}$ are called *local* terms, others are referred to as *external* ones.

Based upon a snapshot of the Semantic Web data collected by Falcons until September 2009, we collect 4,433 Web ontologies, in which most are written in RDF(S) and OWL, while merely a small amount are in DAML+OIL. It is worth noting that, if we define ontologies with respect to separately stored dereferenced documents rather than merging them together, the number of ontologies would be about 25 thousands.

The ontologies contain 2,033,935 local terms in total that cover a lot of real-world domains, e.g., social community, academic publication, music, movie and geography. More specifically, the terms can be classified into 1,895,030 classes and 138,905 properties. A few ontologies have extremely large number of terms, such as YAGO, Cyc, ETHAN, DBpedia and biomedical ontologies FMA, Gene and MeSH. The distribution of the number of terms per ontology is illustrated in Fig. 1, which indicates that the distribution approximates a power law with the exponent $\beta = 1.34$. This power law distribution is in accordance with the observation in [21].

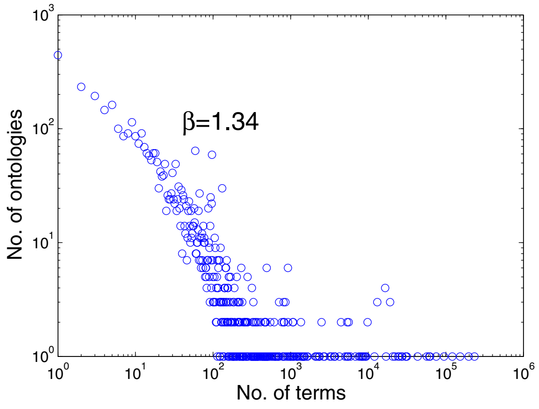


Fig. 1. Power law distribution of the number of ontologies versus the number of terms per ontology

3.2 Experimental Metrics

A *graph* G consists of a finite, non-empty set of *nodes* N and a set of *edges* E. An edge in E is an ordered (for directed graphs) or an unordered (for undirected graphs) pair (u, v), which denotes a connection between $u \in N$ and $v \in N$.

A *weakly/strongly connected component* of G is a subgraph in which any two nodes can be reachable to each other through undirected/directed paths, and to which no more nodes or edges can be added while still preserving its reachability. The number of nodes in a connected component is called its *size*.

The *average distance* for a connected graph is measured as the average shortest path lengths between all the nodes in it. The local clustering coefficient [22] for a node in a connected graph quantifies how close its neighbors are to be a clique (complete graph), and the *clustering coefficient* for the graph is the average of the local clustering coefficients of all nodes. A graph exhibits the *small world* phenomenon, if its clustering coefficient is significantly higher than that of a random graph on the same node set, and if the graph has a short average distance.

A random variable x is distributed according to a *power law* when its probability density function $p(x)$ is in the form of $p(x) = \alpha x^{-\beta}$, where α, β are positive constants, and β is called the *power law exponent*. Power law functions are scale-free, in the sense that if x is re-scaled by multiplying it by a constant, $p(x)$ would still be proportional to $x^{-\beta}$ [18]. According to [7], β can be estimated based on a maximum-likelihood method as follows:

$$\beta \approx 1 + n \left[\sum_{i=1}^{n} \ln \frac{x_i}{x_{\min}} \right]^{-1} . \tag{1}$$

3.3 Ontology Matching and Falcon-AO

Ontology matching (also called mapping or aligning) aims at creating mappings (also known as alignments, correspondences or matches) between semantically matchable terms from different ontologies [10]. In this paper, we define that a *term mapping* is constituted by two terms that hold an equivalence relation, and the matchability between them is in $(0, 1]$ range.

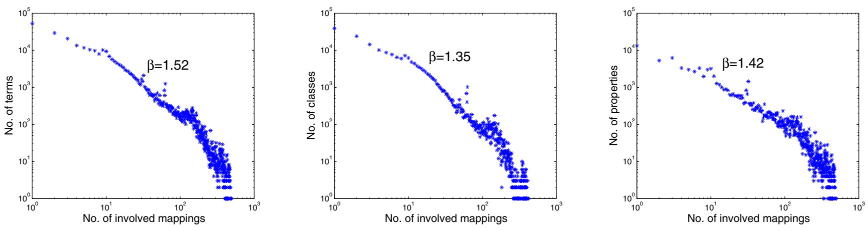
Falcon-AO [16] is a generic, automatic OM tool, which accepts as input two ontologies to be matched, and supplies a library of the edit-distance based and TF-IDF based matchers, the similarity propagating matcher and the partition-based block matcher for large ontologies. Falcon-AO was one of the best tools in all kinds of tests in the OAEI campaign from 2005 to 2007, including the Benchmark, Conference, Directory, Anatomy, Food and Library tracks. Besides good performance, the reasons for selecting Falcon-AO in our experiment include: (i) Falcon-AO is scalable, which is feasible to match very large ontologies; (ii) it is open source. We can easily fix exceptions/bugs during matching; and (iii) it can be run in a batch mode.

4 Term Mapping Graph Analysis

We employ six personal computers and spend nearly one year to pairwise match those 4,433 ontologies, which is a time-consuming process. We create approximate 6 million term mappings with Falcon-AO. In order to make the following analysis more convincible, we filter the mappings with matchability less than 0.7. According to our past experience in OAEI, the threshold 0.7 indicates that the remained mappings can be of high-precision. We also randomly choose a set of 5,000 mappings and perform manual judgement on them. The average precision is about 0.965. After filtering, we retain 3,099,393 mappings between terms. Some statistical data are as follows.

1. Only 280,733 local terms (195,669 classes and 85,064 properties) are involved in these mappings, which are a small part with respect to the total number of terms (2,033,935) in all the ontologies. It indicates that 86.2% terms are unique on the Semantic Web.
2. The number of mappings between classes is 1,553,740 and the number between properties is 1,545,653. In average, a class participates in about 7.9 mappings, while a property is in 18.2 mappings, indicating that properties are more matchable than classes.
3. 45.6% (1,414,406) mappings involve terms with different local names. The local name of a term is a string after the last hash "#" or slash "/" of its URI. There exist one-to-many mappings even within a pair of ontologies, namely, one term in one ontology might be matchable with more than one terms in the other ontology. We totally find 69,457 terms that participate in one-to-many mappings.

Based upon the 3.1 million mappings, we can establish a term mapping graph, where edges are derived from the mappings and nodes are from the terms involved in these mappings. The term mapping graph is undirected, because the mappings that we generate are symmetric. In addition, classes are only matched with classes while properties are matched with properties, so we separately construct a class mapping graph and a property mapping graph.



(a) Terms (b) Classes (c) Properties

Fig. 2. Power law distributions of the number of terms, classes and properties versus the number of involved mappings per term, class and property

Fig. 2(a) shows that the distribution of the number of terms versus the number of involved mappings per term approximates a power law with the exponent $\beta = 1.52$. The distribution does not depict a long tail, which indicates the moderate number of mappings for each term, in other words, no term matches an extremely large amount of other terms. Analogously, the distributions for classes (see Fig. 2(b)) and properties (see Fig. 2(c)) also approximate power laws without long tails. Due to the decentralized nature of the Semantic Web, everyone can publish their own ontologies, which results in many heterogenous definitions of common terms that constitute mappings. As time goes on, some collections of well-defined terms outperform other ones and are universally accepted. So, nonexistence of the long tail reveals the evolution process of ontologies on the Semantic Web.

Fig. 3(a) and 3(b) illustrate the distributions of the number of weakly connected components versus the size of weakly connected component for classes and properties, respectively. Most of these weakly connected components, excluding the largest ones, have sizes less than 200. In addition, the clustering coefficient of the largest weakly connected component for classes is 0.601 and for properties is 0.719, while the average distance for classes is 19.28 and for properties is 8.81, which demonstrate that the property mapping graph forms a small world, however the class mapping graph does not. The average distance for classes is larger than that for properties, which is in accordance with the result that the mappings between classes are sparser than those between properties.

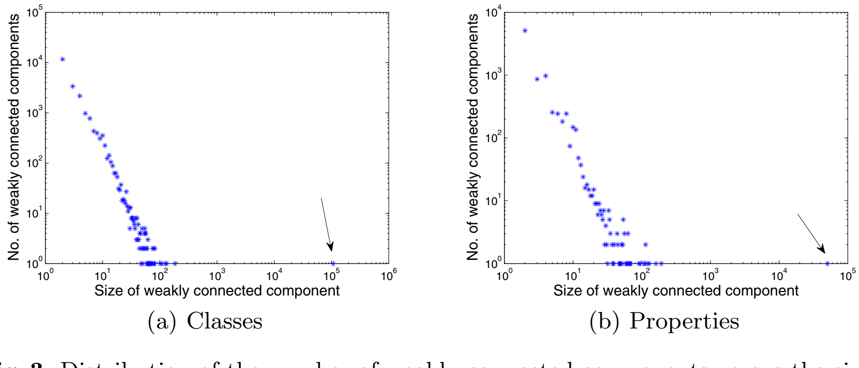


Fig. 3. Distribution of the number of weakly connected components versus the size of weakly connected component

Most weakly connected components have moderate sizes, but the two largest weakly connected components for classes and properties are so large that we need to conduct a deeper investigation. We find that, in contrast to the small weakly connected components, the matchable terms in the largest ones are not so equivalent to each other. This phenomenon can be interpreted as a result of mapping composition [19], caused by ontology or term characteristics deviating in meaning. Then, after merging mappings into a graph, several clusters different in semantics are bridged by some unreasonable mappings, resulting in huge weakly connected components. Therefore, it gives a lesson that we cannot heavily believe in the mapping chains between terms, especially when the transitive chains are long, due to wrong term mapping composition.

Furthermore, we also investigate the most popular local names for classes or properties. We extract the local name of each class or property in our mappings, and count the times of each local name appears (by ignoring string cases). The top-5 local names for classes and properties are listed in Table 1(a) and 1(b), respectively. We see that, although some terms like `foaf:Person` or `dc:title` have been widely accepted, they are still duplicately defined with different URIs in many ontologies. For instance, there are `dbpedia:Person`, `umbel:Person` and many others. These legacy duplications may cause some difficulties in data sharing and reuse, since they weaken the network effect of the Semantic Web.

Table 1. Top-5 popular local names for classes and properties

(a) Classes

	Local name	Times
1	Person	372
2	Organization	254
3	Book	213
4	Article	204
5	Address	179

(b) Properties

	Local name	Times
1	name	468
2	title	278
3	type	237
4	location	237
5	date	205

5 Ontology Mapping Graph Analysis

In this section, we firstly describe the notion of directed edges between ontologies, and then analyze the macroscopic properties of the derived ontology mapping graph.

5.1 Construction of Edges between Ontologies

Each node in an ontology mapping graph is derived from an ontology, while each directed edge is from a set of term mappings between two ontologies. Because the transitivity of matchability in a term mapping graph may cause problems, constructing an edge between two ontologies just depends on those mappings between the terms located in the two ontologies. In other words, an edge exists between two ontologies iff there are explicit term mappings between them.

Although the mappings between terms are undirected, edges between ontologies need a more proper definition, since an ontology contains a collection of terms. Assuming that we have created several mappings between two ontologies, e.g., Food and Pizza. The number of terms involved in the mappings is nearly the same, but considering people's intuitions the matchability is not symmetric especially when noticing the disparity in the sizes of these two ontologies. We may say that Pizza is more matchable to Food whereas the matchability decreases in the other direction. This is supported by the Tversky contrast model

[20], which proposed to compute asymmetric matchability by taking into account both common and different “features” of the things being compared.

Based on the intuition mentioned above and also inspired by [12], we propose a percent-normalized directed edge between two ontologies, which considers not only the term mappings between ontologies, but also the sizes of ontologies for direction and normalization.

Let $\mathcal{O}$ be the source ontology and $\mathcal{O}'$ be the target ontology. $\mathcal{M}^{\gamma}_{\mathcal{O},\mathcal{O}'}$ denotes a set of mappings between the terms in $\mathcal{O}, \mathcal{O}'$ holding their matchability $\geq \gamma$, where $\gamma \in [0.7, 1)$. $\mathcal{T}(\mathcal{O})$ denotes all the local terms in $\mathcal{O}$. $\mathcal{I}(\mathcal{O}, \mathcal{M}^{\gamma}_{\mathcal{O},\mathcal{O}'}) = \{t \in \mathcal{T}(\mathcal{O}) \mid \exists \langle t, t' \rangle \in \mathcal{M}^{\gamma}_{\mathcal{O},\mathcal{O}'}\}$, representing the local terms in $\mathcal{O}$ that are *involved* in $\mathcal{M}^{\gamma}_{\mathcal{O},\mathcal{O}'}$. A directed edge $e^{\gamma,q}_d$ from $\mathcal{O}$ to $\mathcal{O}'$ exists iff $match^{\gamma}_d(\mathcal{O}, \mathcal{O}') > q$, where $q \in [0, 1)$ and $match^{\gamma}_d()$ is defined for measuring how big the overlaps of terms between two ontologies:

$$match^{\gamma}_d(\mathcal{O}, \mathcal{O}') \triangleq \frac{|\mathcal{I}(\mathcal{O}, \mathcal{M}^{\gamma}_{\mathcal{O},\mathcal{O}'})|}{|\mathcal{T}(\mathcal{O})|}. \quad (2)$$

The directed percent-normalized edges reveal how significantly a set of term mappings affect the matchability between ontologies. By changing q, we analyze the characteristics of ontology mapping graph.

5.2 Results

Fig. 4 depicts the variation of the number of edges for different values of q. More specifically, when $q = 0$, the created ontology mapping graph has 1,618,330 directed edges. But the number of edges sharply falls with the increase of q. As compared with the number of ontology pairs, i.e., 9,823,528 ($4433^2/2$), the quantity of edges is quite rare. However, if we realize that the ontologies are collected from the Web and diverse in domains, this result makes sense.

Fig. 5 illustrates the variation of several graph features with the values of q changing from 0.0 to 0.9 for the ontology mapping graph, including the number of connected (not isolated) ontologies, the number of weakly connected components,

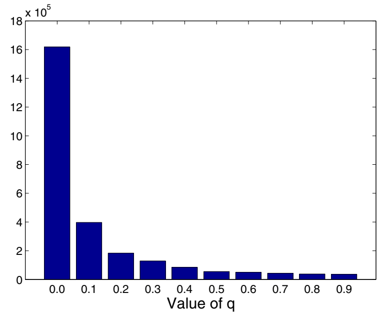


Fig. 4. Variation of the number of directed edges with different values of q

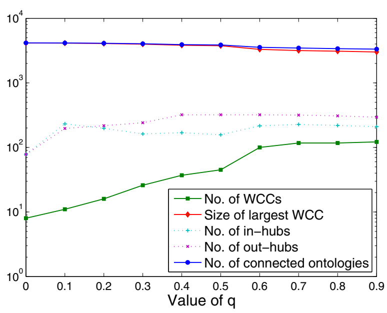


Fig. 5. Variation of graph features with different values of q

the size of the largest weakly connected component and the number of hubs. Here we focus on two kinds of hubs: (i) an in-hub has more than twice the average number of incoming edges of nodes, and (ii) an out-hub has more than twice the average number of outgoing edges of nodes.

With the increase of q, there are more and more isolated ontologies and the size of the largest weakly connected component is shrinking. But the changes are not very sharp and the weakly connected component is almost as large as the whole ontology mapping graph, which indicates a very strong matchability between ontologies. Besides, the proportion of in/out-hubs almost keeps at the level of 10%, which reveals that a small portion of ontologies take part in the connectivity of ontology mapping graph. For $q = 0.1$, the clustering coefficient of the largest weakly connected component is 0.403, while the average distance between the ontologies in it is 2.3, which forms a small world. But, this distance is larger than the one (1.1) in [12], indicating that the average distance between the ontologies in our weakly connected component is longer than that of the particular biomedical domain.

Under a higher threshold value $q = 0.95$ for determining two ontology are matchable or not, we also observe two interesting phenomena. One is the versional evolution which produces a series of versions with slight changes for the same ontology, while the other is the duplicate deployment of the same ontology under different namespaces. Moreover, these two cases often mix with each other. For example, we find two different versions of the Pizza ontology whose version ID changes from 1.1 to 1.4. This ontology is also copied with dozens of different namespaces.

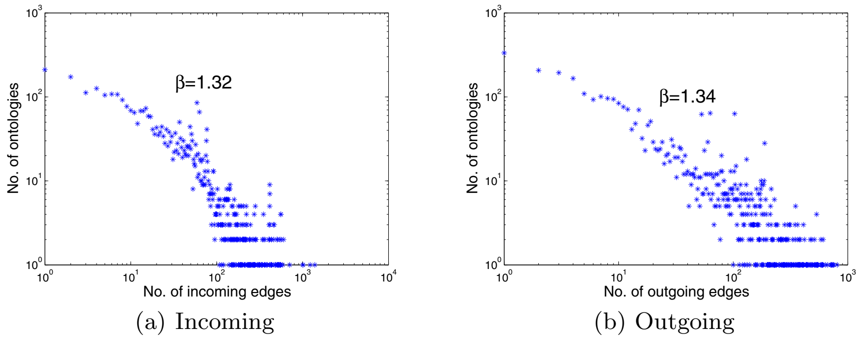


Fig. 6. Power law distribution of the number of ontologies versus the number of incoming/outgoing edges per ontology under $q = 0.2$

Due to space limitation, we merely show here the distributions of the number of incoming and outgoing edges per ontology under $q = 0.2$ in Fig. 6(a) and 6(b), respectively. Both the distributions follow power laws. It is interesting to note that, when q is set to other values (e.g., 0.1, 0.3 or 0.4), the distributions still approximate power laws. Such scale-free nature tells that a few prominent ontologies dominate the connectively of the ontology mapping graph.

The top-5 ontologies with most incoming and outgoing edges for $q = 0.2$ are listed in Table 2 and 3, respectively. Referring to the number of terms contained by the ontologies, in-hubs are usually those ontologies large in size while out-hubs are usually those ontologies small in size. We also observe that in-hubs usually represent common knowledge bases such as DBpedia or prominent ontologies in the mature domains on the Semantic Web, e.g., DCD from the field of biomedicine. [12] indicates that hubs with many outgoing edges show shared domains, in particular at high threshold values for q. However, this method is ineffective for identifying shared domains on the whole Web, because ontologies from the Semantic Web have a great diversity in their sizes and other fields are not as mature as the biomedicine field.

Table 2. Top-5 ontologies with most incoming edges under $q = 0.2$

	URI	#Edges	#Terms
1	`http://dbpedia.org/property/`	1389	24215
2	`http://www.cs.umbc.edu/~aks1/ontosem.owl#`	1228	8501
3	`http://athena.ics.forth.gr:9090/RDF/.../DCD100.rdf#`	1008	5354
4	`http://dbpedia.org/ontology/`	706	889
5	`http://counterterror.mindswap.org/2005/terrorism.owl#`	602	501

Table 3. Top-5 ontologies with most outgoing edges under $q = 0.2$

	URI	#Edges	#Terms
1	`http://vistology.com/ont/bug/error/owl/person.owl#`	809	5
2	`http://www.vistology.com/ont/tests/student4.owl`	757	5
3	`http://tbc.sk/RDF/entity.rdf#`	746	5
4	`http://vistology.com/ont/tests/owlError1.owl#`	737	4
5	`http://vistology.com/.../similarUnused/.../person.owl#`	724	5

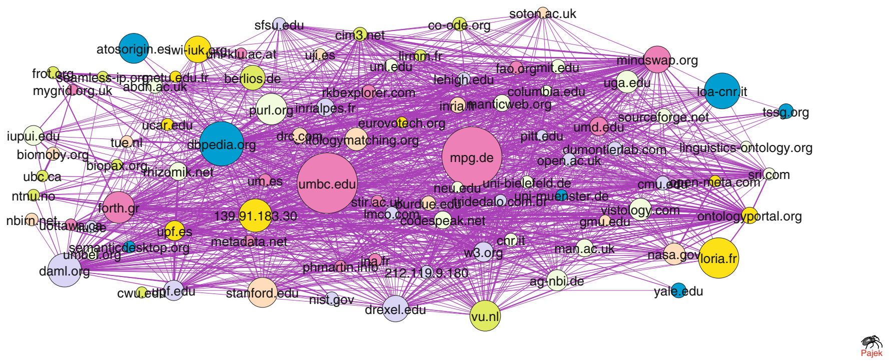
6 Pay-Level-Domain Mapping Graph Analysis

During matching ontologies on the Semantic Web, some common interests between different ontology publishers can be distilled. To reveal relations between these publishers, we introduce the pay-level-domain mapping graph to categorize ontologies into different pay-level-domains and identify their relations.

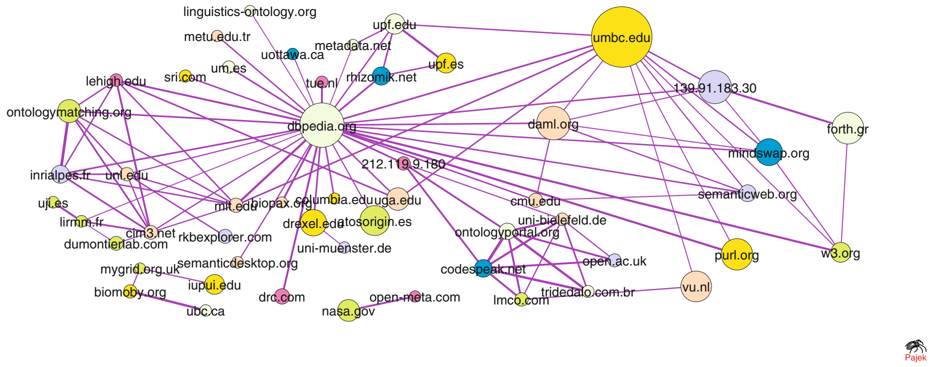
A pay-level-domain mapping graph is defined as an undirected graph, where each node denotes a pay-level-domain that is constituted by the ontologies belonging to it, while each edge denotes a relation between two domains. Let $\mathcal{D}, \mathcal{D}'$ be two pay-level-domains. $\mathcal{M}^{\eta}_{\mathcal{D},\mathcal{D}'}$ denotes a set of mappings among the ontologies in $\mathcal{D}, \mathcal{D}'$ with their matchability $\geq \eta$, where $\eta \in [0, 1)$. $\mathcal{O}(\mathcal{D})$ gives all the ontologies in $\mathcal{D}$. $\mathcal{J}(\mathcal{D}, \mathcal{M}^{\eta}_{\mathcal{D},\mathcal{D}'}) = \{o \in \mathcal{O}(\mathcal{D}) \mid \exists \langle o, o' \rangle \in \mathcal{M}^{\eta}_{\mathcal{D},\mathcal{D}'}\}$, denoting the ontologies in $\mathcal{D}$ that are involved in $\mathcal{M}^{\eta}_{\mathcal{D},\mathcal{D}'}$. An undirected edge $e^{\eta,p}_{u}$ between

$\mathcal{D}, \mathcal{D}'$ exists iff $match_u^\eta(\mathcal{D}, \mathcal{D}') > p$, where $p \in [0, 1)$ and $match_u^\eta()$ is defined to indicate how big the overlaps of ontologies between two publishers:

$$match_u^\eta(\mathcal{D}, \mathcal{D}') \triangleq \min(\frac{|\mathcal{J}(\mathcal{D}, \mathcal{M}^\eta_{\mathcal{D},\mathcal{D}'})|}{|\mathcal{O}(\mathcal{D})|}, \frac{|\mathcal{J}(\mathcal{D}', \mathcal{M}^\eta_{\mathcal{D},\mathcal{D}'})|}{|\mathcal{O}(\mathcal{D}')|}). \tag{3}$$



(a) $p = 0$



(b) $p = 0.3$

Fig. 7. Pay-level-domain mapping graphs under two different values of p (only top-100 pay-level-domains are shown with respect to the number of terms per domain)

We use Nutch[3] to obtain 395 pay-level-domains for the 4,433 ontologies, and select ontology mappings that hold their matchability greater than 0.2 to avoid the influence of "noisy" mappings. Under $p = 0$, the pay-level-domain mapping graph generated from Pajek[4] using the top-100 biggest pay-level-domains with respect to the number of terms in each domain is depicted in Fig. 7(a). There are 90 domains matchable with each other, while the left 10 domains have no

3 http://nutch.apache.org/

4 http://vlado.fmf.uni-lj.si/pub/networks/pajek/

connection with others, since the ontology mappings for these domains have low matchability and are filtered before. There is only one big connected component in the figure, and the mappings between these domains are very complex, which means that most ontology publishers connect with each other more or less.

We increase p to 0.3 in order to filter some insignificant edges. Many edges are omitted and only 95 edges left. As a result, 38 domains are removed since all edges linking to them are deleted. A clearer depiction is shown in Fig. 7(b), where `DBpedia.org` is the biggest hub in this pay-level-domain mapping graph and `umbc.edu` ranks the second. Several active organizations on the Semantic Web, e.g., UMBC, W3C and DAML, have already provided a large amount of ontologies, which are matched with other ones. In view of `DBpedia.org`, data producers are welcome to link their data into the Linked Data cloud, thus `DBpedia.org` becomes a central point on the Semantic Web. Besides, two small connected components are separated, where one contains two domains (`nasa.gov` and `open-meta.com`), and the other contains four domains about biomedicine.

The visualization of the pay-level-domain mapping graph provides insights into how publishers are connected. Both Fig. 7(a) and 7(b) show a picture of how publishers share their interests. Note that pay-level-domain mapping graph is based on ontology mapping graph, we can conclude that most publishers are interested in a diversity of topics, which is demonstrated in Fig. 7(a), while deep interests can be seen from a specified pay-level-domain mapping graph (e.g., Fig. 7(b)).

7 Conclusion

In this paper, we collect more than 4 thousand ontologies based on the Falcons search engine, and perform a large-scale pairwise matching with a scalable ontology matching tool Falcon-AO. We generate 3.1 million term mappings, which are used for analyzing the morphology of term mapping graph, ontology mapping graph and pay-level-domain mapping graph. To the best of our knowledge, our work is the first attempt towards analyzing the matchability between such a large number of ontologies on the Semantic Web, where the difficulties lie in the high computational cost and the messiness of real Semantic Web data.

By analysis, we make some observations as follows. Firstly, both the term mapping graph and ontology mapping graph inherit some characteristics of the Hypertext Web and Semantic Web, such as the scale-free nature and the small world. Secondly, a small portion of terms are well matched, while many cannot match any others, which demonstrate the skewed matchability between terms. However, most ontologies are loosely connected to each other. Thirdly, ontology publishers show common interests in ontology development, where `DBpedia.org` and `umbc.edu` are the two most active publishers. Lastly, our experimental results confirm some existing conclusions on a small set of ontologies, but we also find some differences. For example, the average distance between our ontologies is twice larger than the one in the biomedical domain.

From a practical viewpoint, our downloadable term mappings can help both ontology developers and users in the process of ontology creation, integration and maintenance. For example, before creating an ontology, developers could check if similar terms (e.g., `foaf:Person`, `dc:title`) or ontologies have already been defined. Even if ontologies were created, people can still link their terms with popular ones based on our mappings, by using `owl:equivalentClass` and `owl:seeAlso` constructs to gain potential interoperability. We believe that, for some domains, reusing well-known ontologies and terms rather than "reinventing the wheel" would facilitate data integration and sharing for a better Data Web; while for other domains, more efforts are expected to create new ontologies or synthesize existing ones. Another example is when conducting ontology matching or data fusion, users can identify representative hubs (e.g., DBpedia, SUMO and OpenGALEN) as useful background knowledge.

The analytic results reported in this paper is just the first step, and many issues still need to be addressed further. In the near future, we look forward to using other robust and scalable ontology matching tools to repeat some part of the experiment and confirm our observations. Another important problem raised from our study is how to utilize these mappings for potential applications, such as object consolidation and Semantic Web data browsing.

Acknowledgements

This work is supported in part by the NSFC under Grant 61003018, 60973024 and 61021062, and also in part by the National Research Foundation for the Doctoral Program of Higher Education of China under Grant 20100091120041. We are grateful to Gong Cheng and Weiyi Ge for providing ontologies in the Falcons search engine. We also appreciate the helpful comments from three anonymous reviewers.

References

1. Adamic, L., Huberman, B.: Power-Law Distribution of the World Wide Web. Science 287(5461), 2115a (2000)
2. Albert, R., Jeong, H., Barabasi, A.: The Diameter of the World Wide Web. Nature 401, 130–131 (1999)
3. Berrueta, D., Phipps, J.: Best Practice Recipes for Publishing RDF Vocabularies. W3C Working Group Note (2008)
4. Bizer, C., Schultz, A.: The R2R Framework: Publishing and Discovering Mappings on the Web. In: ISWC Workshop on Consuming Linked Data (2010)
5. Broder, A., Kumar, R., Maghoul, F., Raghavan, P., Rajagopalan, S., Stata, R., Tomkins, A., Wiener, J.: Graph Structure in the Web. Computer Networks 33(1-6), 309–320 (2000)
6. Cheng, G., Qu, Y.: Searching Linked Objects with Falcons: Approach, Implementation and Evaluation. International Journal on Semantic Web and Information Systems 5(3), 49–70 (2009)
7. Clauset, A., Shalizi, C., Newman, M.: Power-Law Distributions in Empirical Data. SIAM Review 51(4), 661–703 (2009)

8. Ding, L., Shinavier, J., Shangguan, Z., McGuinness, D.: SameAs Networks and Beyond: Analyzing Deployment Status and Implications of owl:sameAs in 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. 145–160. Springer, Heidelberg (2010)
9. Donato, D., Laura, L., Leonardi, S., Millozzi, S.: The Web as a Graph: How Far We Are. ACM Transactions on Internet Technology 7(1), 1–25 (2007)
10. Euzenat, J., Shvaiko, P.: Ontology Matching. Springer, Heidelberg (2007)
11. Ge, W., Chen, J., Hu, W., Qu, Y.: Object Link Structure in the Semantic Web. In: Aroyo, L., Antoniou, G., Hyvönen, E., ten Teije, A., Stuckenschmidt, H., Cabral, L., Tudorache, T. (eds.) ESWC 2010. LNCS, vol. 6089, pp. 257–271. Springer, Heidelberg (2010)
12. Ghazvinian, A., Noy, N., Jonquet, C., Shah, N., Musen, M.: What Four Million Mappings Can Tell You about Two Hundred Ontologies. In: Bernstein, A., Karger, D.R., Heath, T., Feigenbaum, L., Maynard, D., Motta, E., Thirunarayan, K. (eds.) ISWC 2009. LNCS, vol. 5823, pp. 229–242. Springer, Heidelberg (2009)
13. Gil, R., Garcia, R., Delgado, J.: Measuring the Semantic Web. AIS SIGSEMIS Bulletin, 69–72 (2004)
14. Halpin, H., Hayes, P.J., McCusker, J.P., McGuinness, D.L., Thompson, H.S.: When owl:sameAs Isn't the Same: An Analysis of Identity in 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. 305–320. Springer, Heidelberg (2010)
15. Hoser, B., Hotho, A., Jäschke, R., Schmitz, C., Stumme, G.: Semantic Network Analysis of Ontologies. In: Sure, Y., Domingue, J. (eds.) ESWC 2006. LNCS, vol. 4011, pp. 514–529. Springer, Heidelberg (2006)
16. Hu, W., Qu, Y.: Falcon-AO: A Practical Ontology Matching System. Web Semantics: Science, Services and Agents on the World Wide Web 6(3), 237–239 (2008)
17. Nikolov, A., Motta, E.: Capturing Emerging Relations Between Schema Ontologies on the Web of Data. In: ISWC Workshop on Consuming Linked Data (2010)
18. Theoharis, Y., Tzitzikas, Y., Kotzinos, D., Christophides, V.: On Graph Features of Semantic Web Schemas. IEEE Transcations on Knowledge and Data Engineering 20(5), 692–702 (2008)
19. Tordai, A., Ghazvinian, A., van Ossenbruggen, J., Musen, M., Noy, N.: Lost in Translation? Empirical Analysis of Mapping Compositions for Large Ontologies. In: ISWC Workshop on Ontology Matching (2010)
20. Tversky, A.: Features of Similarity. Psychological Review 84(4), 327–352 (1977)
21. 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.) ASWC 2007 and ISWC 2007. LNCS, vol. 4825, pp. 552–565. Springer, Heidelberg (2007)
22. Watts, D., Strogatz, S.: Collective Dynamics of 'Small-World Networks'. Nature 393(6684), 440–442 (1998)

Understanding an Ontology through Divergent Exploration

Kouji Kozaki, Takeru Hirota, and Riichiro Mizoguchi

The Institute of Scientific and Industrial Research, Osaka University
8-1 Mihogaoka, Ibaraki, Osaka, 567-0047 Japan
{kozaki,hirota,miz}@ei.sanken.osaka-u.ac.jp

Abstract. It is important that the ontology captures the essential conceptual structure of the target world as generally as possible. However, such ontologies are sometimes regarded as weak and shallow by domain experts because they often want to understand the target world from the domain-specific viewpoints in which they are interested. Therefore, it is highly desirable to have not only knowledge structuring from the general perspective but also from the domain-specific and multi-perspective so that concepts are structured for appropriate understanding from the multiple experts. On the basis of this observation, the authors propose a novel approach, called divergent exploration of an ontology, to bridge the gap between ontologies and domain experts. Based on the approach, we developed an ontology exploration tool and evaluated the system through an experimental use by experts in an environmental domain. As a result, we confirmed that the tool supports experts to obtain meaningful knowledge for them through the divergent exploration and it contributes to integrated understanding of the ontology and its target domain.

Keywords: ontology, divergent exploration, view point, conceptual map.

1 Introduction

Ontologies are designed to provide underlying conceptual structure and machine readable vocabulary of domains for Semantic Web applications. Ontology is defined as “An explicit specification of conceptualization”[1], and it clearly represents how the target world is captured by people and systems. That is, an ontology construction implies to understand the target world, and understanding the ontology means understanding the target world to some extent. Especially, an ontology plays an important role for comprehensive understanding of a complex domain which consists of many sub-domains. To use an ontology for this purpose, there are two approaches: 1) mapping ontologies which are built for each domain, and 2) building domain ontlogies based on the same shared upper ontology. While the former is easily acceptable to domain experts, ontology mapping is a hard task and needs large cost. To avoid the mapping issue, many research groups take the latter approach. OBO Foundry coordinates activities to develop and share ontologies which cover common concepts across domains[2]. That is, knowledge sharing and exchanging across domains can be realized through the shared common vocabulary defined in the same ontology.

© Springer-Verlag Berlin Heidelberg 2011

For knowledge sharing through an ontology, it is important that the ontology captures the essential conceptual structure of the target world as generally as possible and they should be well organized with consistency and reusability. Even if it is domain-dependent and specialized concept, its meaning could be shared with people working in other domains through its definition in terms of generalized common concepts. For example, when we define a specialized relationship well-known in a domain, it could be represented as combinations of general relationships by decomposing the original relation. The generalized representation makes implicit knowledge explicit and machine readable one. In this way, an ontology contributes to readability and interoperability of knowledge through domain-independent and generalized conceptualization. However, such generalized ontologies are sometimes regarded as verbose and shallow by domain experts because they often want to understand the target world from the domain-specific viewpoints in which they are interested. In many cases their interests are different and versatile, even if they are experts in the same domain. It is a serious and important issue to bridge the gap between ontologies which try to cover wide area domain-independently and interests of domain experts which are well-focused and deep. Therefore, it is highly desirable to have not only knowledge structuring from the general perspective but also from the domain-specific and multi-perspective so that concepts are structured for appropriate understanding from the multiple experts.

On the basis of this observation, the authors propose a novel approach to bridge the gap between ontologies and domain experts. The main strategy is composed of: (1) the conceptual structure of an ontology is systematized as generally as possible and (2) on the fly reorganizing some conceptual structures from the ontology as visualizations to cope with various viewpoints which reflects interests of the domain experts (Fig.1). Based on this strategy, we developed a frame work, named divergent exploration of ontology, and an ontology exploration tool as implementation of it. The tool allows users to explore an ontology according to their own perspectives and visualizes them in a user-friendly form, i.e. conceptual map. It contributes to helping users explore the ontology from several viewpoints to eventually obtain integrated understanding of the ontology and its target domain. Furthermore, it stimulates their intellectual interest and could support idea creation.

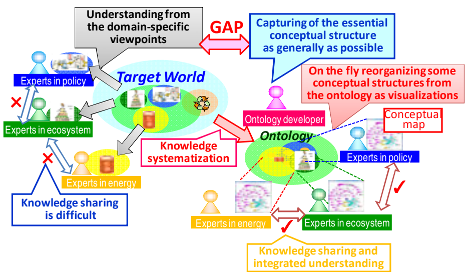

Understanding from the domain-specific viewpoints
GAP
Capturing of the essential conceptual structure as generally as possible
Target World
Experts in policy
Experts in ecosystem
Experts in energy
Knowledge sharing is difficult
On the fly reorganizing some conceptual structures from the ontology as visualizations
Ontology developer
Conceptual map
Ontology
Knowledge systematization
Experts in policy
Experts in energy
Experts in ecosystem
Knowledge sharing and integrated understanding

Fig. 1. Integrated understanding of cross-domain knowledge through divergent exploration of ontology

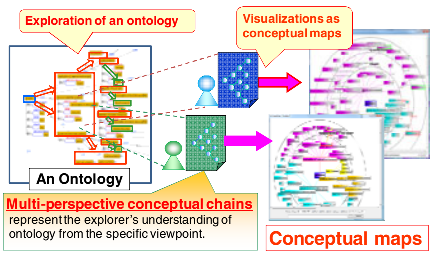

Exploration of an ontology
Visualizations as conceptual maps
An Ontology
Multi-perspective conceptual chains
represent the explorer's understanding of ontology from the specific viewpoint.
Conceptual maps

Fig. 2. Divergent exploration of ontology

The rest of this paper is organized as follows: In section 2, we introduce divergent exploration for understanding an ontology from multi-perspectives. After we discuss requirements for the ontology exploration tool, we discuss its functionality in section 3. In Section 4, we evaluate the system through its application to an environmental domain and an experimental use by domain experts. In section 5, we summarize some related works. Finally, we present concluding remarks with future work.

2 Divergent Exploration

2.1 Divergent Exploration of an Ontology

Most of semantic web applications use ontologies as vocabularies to describe meta-data and are aimed at semantic processing of them. By contrast, we regard ontology as the target for divergent exploration of the ontology itself. The divergent exploration of an ontology enables users to explore a sea of concepts in the ontology freely from a variety of perspectives according to their own motive. The exploration stimulates their way of thinking and contributes to deeper understanding of the ontology and hence its target world. As a result, the users can find out what they take interest. Some of them could include new findings for them because they could obtain unexpected conceptual chains which they have never thought through the ontology exploration.

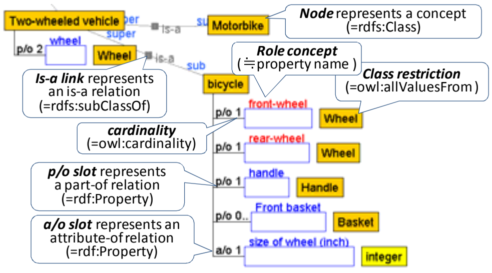

Two-wheeled vehicle
super
is-a
sub
Motorbike
Node represents a concept (=rdfs:Class)
p/o 2
wheel
super
Wheel
is-a
sub
Role concept (≒property name)
Is-a link represents an is-a relation (=rdfs:subClassOf)
bicycle
Class restriction (=owl:allValuesFrom)
p/o 1
front-wheel
Wheel
cardinality (=owl:cardinality)
p/o 1
rear-wheel
Wheel
p/o slot represents a part-of relation (=rdf:Property)
p/o 1
handle
Handle
p/o 0..
Front basket
Basket
a/o slot represents an attribute-of relation (=rdf:Property)
a/o 1
size of wheel (inch)
integer

Fig. 3. An example ontology representation in Hozo and its correspondence with OWL

Table 1. The aspects of ontology exploration for each relationship

	Related relationships		Kinds of extraction
	Hozo	OWL	
(A)	*is-a* relationship	rdfs:subClassOf	(1) Extraction of sub concepts
			(2) Extraction of super concepts
(B)	*part-of / attribute-of* relationship	rdf:properties which are referred in	(3) Extraction of concepts referring to other concepts via relationships
			(4) Extraction of concepts to be referred to by some relationship
(C)	*depending-on* relationship		(5) Extraction of contexts
			(6) Extraction of role concepts
(D)	*play (playing)* relationship		(7) Extraction of players
			(8) Extraction of role concepts

Fig.2 outlines the framework of ontology exploration. The divergent exploration of an ontology can be performed by choosing arbitrary concepts from which, according to the explorer's intention, they trace what we call *multi-perspective conceptual chains*. We define the viewpoint for exploring an ontology and obtaining the multi-perspective conceptual chains as the combination of a *focal point* and *aspects*. The focal point indicates a concept to which the user pays attention as a starting point of the exploration. The aspect is the manner in which the user explores the ontology. Because an ontology consists of concepts and the relationships among them, the aspect can be represented by a set of methods for extracting concepts according to its relationships. The multi-perspective conceptual chains are visualized in a user-friendly form, i.e., in a conceptual map. Although there are many researches

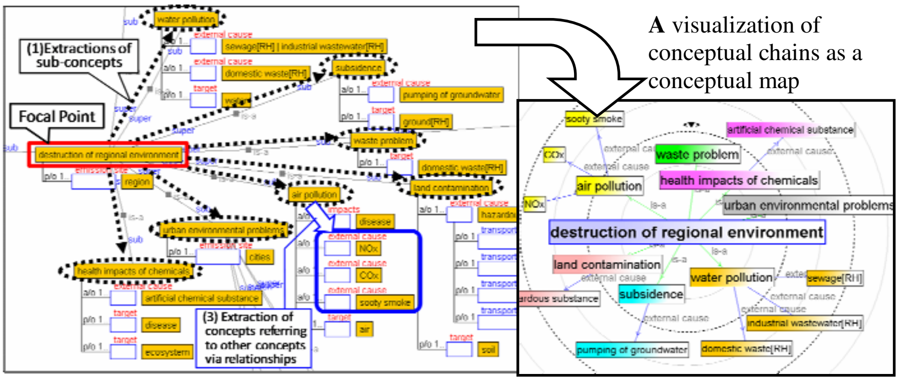

(1)Extractions of sub-concepts
Focal Point
destruction of regional environment
(3) Extraction of concepts referring to other concepts via relationships
A visualization of conceptual chains as a conceptual map
water pollution
air pollution
waste problem
land contamination
subsidence
urban environmental problems
health impacts of chemicals
sooty smoke
NOx
COx
artificial chemical substance
pumping of groundwater
domestic waste[RH]
industrial wastewater[RH]
sewage[RH]

Fig. 4. An example of divergent exploration of an ontology

on visualization of ontology, the main purpose of our research is not the visualization itself but exploration of an ontology. It is neither ontology browsing which are supported by most of ontology development tool nor ontology summarization. The divergent exploration of an ontology aims at integrated understanding of the ontology and its target world from multiple perspectives across domains according to the users' interests. We also focus on that our tool supports domain experts to obtain meaningful knowledge for themselves as conceptual chains through the divergent exploration.

2.2 Definition of View Points

We implement the divergent exploration of an ontology as an additional function of Hozo which is our ontology development tool[3]. Fig.3 shows an example of ontology defined using Hozo. Ontologies are represented by nodes, slots and links. The nodes represent concepts (classes), *is-a* links represent *is-a* (subclass-of) relations, and slots represents *part-of* (denoted by "p/o") or *attribute-of* (denoted by "a/o") relations. A slot consists of its kind ("p/o" or "a/o"), *role concept*, *class restriction*, *cardinality*. Roughly speaking, a slot corresponds to *property* in OWL and its role name represent name of property. Its class restriction and cardinality correspond owl:allValuesFrom and owl:cardinality respectively. However, semantics of Hozo's ontology includes some concepts related in role which are not supported in OWL because it is designed based on ontological theory of role[4]. While we have designed three levels of role representation model in OWL to capture the semantics level-wise [5], we use simplest model described above in this paper.

As described above, viewpoints are defined as a combination of a focal point and aspects, and an aspect is represented according to relationships defined in an ontology. We classify these relationships into four kinds and define two aspects of ontology exploration for each relationship according to the direction to follow (upward or downward) (See Table.1). The user can control kinds of relationships to follow by specifying kinds of role concept (properties) in the aspects type (B) to (D). We call the control "role limitation of aspect". Similarly, users can constrain the types of concepts to reach through aspects by specifying types of concepts. We call the constraint “class type limitation of aspect”. While we can suppose more detailed viewpoints such as limitation of cardinality, we do not introduce them because they are too be detailed for domain experts to explore easily.

Fig.4 shows an example of an ontology exploration. The user set *Destruction of regional environment* as the focal point and select (1) Extraction of sub concepts as an aspect. Then, following *is-a* relations, seven concepts such as *Air pollution*, *Land contamination*, etc. are extracted. Next, if the user focus on Air pollution and selects (3) Extraction of concepts referring to other concepts via relationships as an aspect, *Disease*, *NOx*, *COx*, *Sooty smoke* and *Air* are extracted following attribute-of relations of *Air pollution*. On the other hand, if the user applies external cause as role limitation of aspects, only *NOx*, *COx* and *Sooty smoke*, which are related to external cause, are extracted (Fig. 4 left). As a result of this concept extraction, the system generates conceptual chains that match the user’s interest and visualizes them as a conceptual map. In the conceptual map, extracted concepts and followed relationships are represented as nodes and links respectively, and the nodes are located on concentric circles in which the focal point is located at the center. As a result, the conceptual chains are represented as a divergent network (Fig.4 right). In this way, the user can explore an ontology from various viewpoints by choosing combinations of focal points and aspects, and the results are visualized as conceptual maps.

2.3 Requirements for the Ontology Exploration Tool

In this section, we discuss functions which the ontology exploration tool is required to support. The viewpoints for ontology exploration are defined by combination of focal points and aspects. In many cases, the combination is fixed through repetitions of choice of aspects for the exploration by trial and error because not every user has clear intentions and viewpoints to explore the ontology at first. That is, the user explores the ontology step by step and clarifies his/her interest gradually. The result of this process is represented as multi-perspective conceptual chains. Functions to support such a step by step exploration of the user are required as the most fundamental function of the ontology exploration tool.

On the other hands, it is required to make full use of semantic processing, which is a feature of ontology, for exploration. For example, a function to search all combinations of aspects automatically and get conceptual chains which represents paths from a focused concept to another specified concept, we call it *search path* function, seems to be useful. Functions to investigate obtained conceptual maps from additional perspectives such as *change view* and *comparison* function are also required. The change view is a function to apply additional viewpoints to a conceptual map generated based on another viewpoint, and as a result the visualization of the map is changed according to the specified concepts or relationships in the additional viewpoint. And a comparison function enables the user to compare two conceptual maps and visualize the common elements in them.

3 Implementation of the Ontology Exploration Tool

3.1 System Architecture

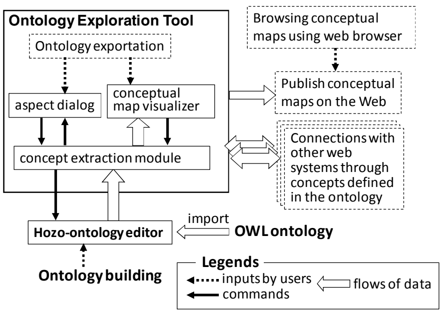

Ontology Exploration Tool
Ontology exportation
aspect dialog
conceptual map visualizer
concept extraction module
Browsing conceptual maps using web browser
Publish conceptual maps on the Web
Connections with other web systems through concepts defined in the ontology
Hozo-ontology editor
import
OWL ontology
Ontology building
Legends
inputs by users
commands
flows of data

Fig. 5. The architecture of ontology exploration tool

Fig.5 shows the architecture of the ontology exploration tool. It consists of *aspect dialog*, *concept extraction module* and *conceptual map visualizer*. The aspect dialog provides graphical user interface to select viewpoints for ontology extraction. The concept extraction module follows relationships between concepts according to the selected viewpoint and obtains multi-perspective conceptual chains. The conceptual chains are visualized as conceptual maps by conceptual map visualizer. While the target of the system is an ontology in Hozo's format, it also can support an ontology in OWL because Hozo can import OWL ontology. The generated conceptual maps can be connected with other web systems through concepts defined in the ontology. For example, mapping nodes in the conceptual map with Linked Data allows the user to access various web resources through it. While the ontology exploration tool is implemented as a client application by Java, it can export generated maps in an XML format and publish them on the Web. Users can browse them using web browsers with a conceptual map viewer implemented by Flash. Demos of browsing the conceptual map and download of the system are available at the URL: http://www.hozo.jp/OntoExplorer/ .

3.2 Functions for Ontology Exploration

The ontology exploration tool provides the following functions (Fig.6).

Detailed exploration using aspect dialog: The aspect dialog lists kinds of aspect with numbers of concept which will be extracted when the aspect is selected. The user selects an aspect with detailed setting for exploration such as role limitation and extracting class type limitation. The user can explore from detailed viewpoint by repetition of selecting aspects with the settings.

Simple exploration: The conceptual map visualizer provides commands to apply some typical combinations of aspects as mouse menus. The user can explore an ontology by simple operation using them. While this operation does not allow detailed settings, we suppose beginners of the tool can use it and explore easily.

Search path (machine exploration): The system can search all combination of aspects to generate conceptual chains from a concept selected as starting point to those specified by the user. As a result, the system shows all conceptual chains between the selected concepts.

Change view: The tool has a function to highlight specified paths of conceptual chains on the generated map according to given viewpoints. For example, when the user specifies a focusing concept, the conceptual map visualizer highlights the paths which include the concepts or its sub concepts in them. It also can display only the highlighted paths on the map by hiding other paths. Another viewpoint for highlighting includes specifying the focusing kinds of relationships (aspects) and specifying concepts or relationships which has some values of attributes. When the user specifies several viewpoints at the same time, the system highlights paths in different colors according to the viewpoints. Through this function, the user can switch several viewpoints easily and compare difference between them. For example, when we suppose a conceptual map which represents effects of global warming, the user can change viewpoints according to time scale which the effects will occur and/or spatial extent of them.

Comparison of maps: The system can compare generated maps and show the common conceptual chains both of the maps.

4 Usage and Evaluation of Ontology Exploration Tool

4.1 Usage for Knowledge Structuring in Sustainability Science

Sustainability science (SS) is a discipline aimed at establishing new disciplinary schemes that serve as a basis for constructing a vision that will lead global society to a sustainable one. Meeting this objective requires an interdisciplinary integrated

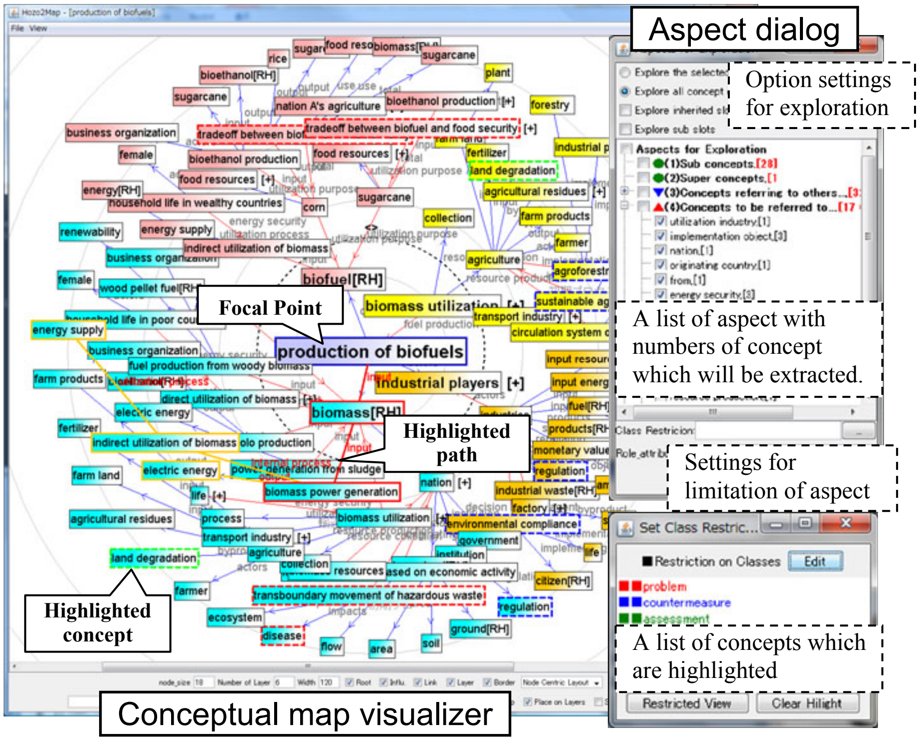

Aspect dialog
Option settings for exploration
A list of aspect with numbers of concept which will be extracted.
Settings for limitation of aspect
A list of concepts which are highlighted
Focal Point
production of biofuels
Highlighted path
Highlighted concept
Conceptual map visualizer
Aspects for Exploration
(1)Sub concepts.[28]
(2)Super concepts.[1
(3)Concepts referring to others...[3
(4)Concepts to be referred to...[17
Set Class Restric...
Restriction on Classes
Edit
problem
countermeasure
assessment
Restricted View
Clear Hilight

Fig. 6. A snapshot of the ontology exploration tool

understanding of the entire field instead of knowledge structuring that depends on individual related domains. Thus, Osaka University Research Institute for Sustainability Science (RISS) has been working on construction of an SS ontology in which knowledge of domains relating to SS is organized into a common domain-independent conceptual structure. Furthermore, with the ontology exploration tool prototype developed in this research, by generating conceptual maps from the SS ontology in accordance with users' viewpoints, attempts were at knowledge structuring in sustainability science, such as knowledge sharing among experts in different domains and an integrated understanding from multiple viewpoints. As a result, through usage of the tool by experts at RISS, it was confirmed that the tool had a certain utility for achieving knowledge structuring in sustainability science [6]. The ontology exploration tool described in Section 3 is a tool for enabling an overview of multiple domains, improved on the basis of the above findings. Furthermore, in this research, we conducted evaluation in a setting closer to that used in practice.

4.2 Verification of Exploring Ability of Ontology Exploration Tool

In order to verify whether the tool can properly explore an ontology and generate conceptual maps that domain experts wish to do, we enriched the SS ontology in a more specific domain (biofuels) and verified the exploring ability of the tool.

4.2.1 Enrichment of SS Ontology

First, we constructed a Biofuel ontology by enriching the SS ontology described in Section 4.1 with concepts relating to biofuels. Before enriching the ontology, from base material collected by reviewing existing research, domain experts organized the structures of target problems in ontology construction into 44 typical scenarios representing instances of how the production and usage of biofuels affect various fields. Each of the scenarios was expressed in the form of a short sentence, such as "(1-3) Biofuels can replace only a small share of the global energy supply, and biofuels alone are not sufficient to overcome our dependence on fossil fuels" or "(2-6) Small-scale labor-intensive bioenergy production is effective in creating employment but has drawbacks with production efficiency and economic competitiveness (tradeoff relationship)." The scenarios are classified into the nine categories listed in Table 2.

Table 2. The numbers of scenarios

Problem category	Number of scenarios
1) Energy services for the poor	3
2) Agriculture and industry development and employment creation	6
3) Health and gender	4
4) Agricultural structures	4
5) Food security	6
6) Government budget	4
7) Trade, foreign exchange balance, energy security	5
8) Biodiversity and natural resource management	8

Then, on the basis of the contents of these scenarios, keeping in mind that the enriched ontology should serve for conceptual map generation, ontology experts took the following procedure:

1) Add main concepts appearing in the scenarios to the ontology.
2) Clarify relationships among the concepts appearing in the contents of the scenarios, including between-the-lines relationships (hidden causal chains) not explicit in the text of the scenarios.
3) Describe the relationships clarified in step 2 as an ontology.

For example, in the case of the scenario "(4-1) Demand for energy crop production increases pressure on land (farms) for food production, resulting in a rise in food prices," concepts added to the ontology in step 1 were Demand for biomass resources, Farms for food production, and Rise in food prices, and the relationship clarified in step 2 was "Increase in demand for biomass resources → Increase in farms for fuel production → Problem of fixed area → Decrease in farms for food production → Decrease in food supply → Rise in food prices." When we added concepts and relationships in ontology, we defined not only them but also upper concepts of them. For example, relationships between farms for fuel production and farms for food production were generalized as a finite total amount. These generalized definitions enable the users to generate a wide variety of conceptual maps which are not directly represented in the scenario.

Through such examination, we reorganized the 44 scenarios listed in Table 1 into 29 scenarios, in consideration of complex relationships and similarity of relationships, and enriched the ontology for the 29 scenarios. The scale of the resulting biofuel ontology was about twice as large as the scale of the SS ontology described in Section 4.1; specifically, the number of concepts increased from 649 to 1892, and the number of slots increased from 1075 to 2119.

4.2.2 Verification of Scenario Reproducing Operation

We verified whether the ontology enriched by the method described above can properly express the original problem structures by examining whether it was possible, with the ontology exploration tool, to generate conceptual maps in which the contents of the original scenarios were reproduced. As exploration (viewpoint setting) methods used for map generation, the following three methods were attempted. As a result, the number of original scenarios for which conceptual maps corresponding to the problem structures represented by the scenarios were successfully generated (reproduced) and the ratio of such scenarios to the total number of scenarios were as follows:

1) Scenarios reproduced by exploration with Search Path function (automatic exploration of combinations of relationships among all the concepts): 21 (72%)
2) Scenarios reproduced by simple exploration (exploration is performed by using only simplified exploration conditions (aspects), and the Change View function for extracting only the paths including specified concepts is used): 24 (82%)
3) Scenarios reproduced by simple exploration or detailed exploration (exploration performed with a selected detailed viewpoint): 27 (93%)

As for the two scenarios that were not reproduced, we found that inadequate ontology definition was the reason for the failure to reproduce conceptual maps corresponding to these scenarios. That is, the two scenarios could not reproduced as conceptual maps not because ability of the tool but because we missed to add some relationships in the two scenarios. In short, we can conclude that the exploration ability and conceptual map expressing ability of the tool were sufficient for reproducing target scenarios in the form of conceptual maps.

4.3 Experiment for Evaluating Ontology Exploration Tool

4.3.1 Experiment Overview

In order to verify that the ontology exploration tool developed in this research is useful for obtaining a domain overview, we conducted an evaluation experiment with cooperation from experts. In the experiment, we used the Biofuel ontology described in Section 4.3. The experiment was aimed at evaluating the following two issues through actual usage of the tool by the experts:

1) Whether meaningful maps that provide intellectual stimulation were obtained.
2) Whether meaningful maps other than those representing the contents of the scenarios anticipated at the time of ontology construction were obtained.

The subjects of the experiments were four experts A to D in different fields of expertise. The experts A and B had no experience of using the tool at all, and the experts C and D had some experience. The fields of expertise of the individual experts were as follows; A: Agricultural economics, B: Social science (stakeholder analysis), C: Risk analysis, D: Metropolitan environmental planning. Although someone may suppose just four persons are too few for evaluation, note that they are neither students nor public users which can be easily collected in usual cases. We believe that the experiments by four real experts are meaningful as an initial evaluation of the tool.

4.3.2 Experimental Method

First, we asked the four experts to generate conceptual maps with the tool in accordance with condition settings of the following tasks 1 to 3.

Task 1: Under the supposed problem "What kinds of *Environmental destruction problems* relating to *Biofuels* exist," conceptual maps were generated under the conditions that the concept that could be selected first (Focal Point) was restricted to *Bioenergy usage* or *Biomass resource production* and the exploration method was either 1) automatic exploration by Search Path function or 2) simple exploration described in Section 4.2.2. At this time, the experts were allowed to freely select concepts used for viewpoint setting with Search Path or Change View from sub concepts of *Environmental destruction problems*.

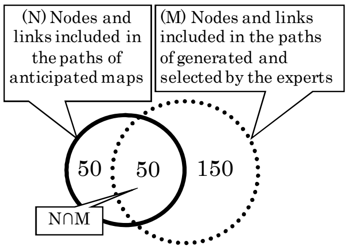
Task 2: "What kinds of *Tradeoff problems* relating to *Biofuels* exit?" We asked the experts to generate conceptual maps under the same conditions as those in Task 1, except that they were allowed to freely select concepts used for viewpoint setting with Search Path or Change View from sub concepts of *Tradeoff problems*.

Task 3: We asked the experts to generate conceptual maps by arbitrary methods regarding problems related to their interests.

Then, from the paths of conceptual chains (visualized paths that track a series of relationships among concepts) included in the conceptual maps generated, paths that were clearly judged as inappropriate by the experts were removed, and evaluation was conducted using the paths selected by the subjects according to their interests. Note here, although we restricted operations for exploration in Task 1 and 2, the number of combinations allowed for the subjects were 184 and 64 respectively at least[1]. Furthermore, they did not know the contents of scenarios which we supposed in each task. Therefore, the selected paths did not always correspond to the scenarios. Then we asked the subjects to enter, via a special input screen on the tool, a four-level general evaluation (A: Interesting, B: Ordinary but important, C: Neither good or poor, D: Obviously wrong), a four-level evaluation (Excellent, Good, Normal, Bad) regarding four specific points (1: Clarity, 2: Faithful reproduction, validity, and appropriateness, 3: Ease of overview, coverage, and comprehensiveness, 4: Conception and discovery assistance (stimulation)), and free comments.

Table 3. Experimental results

		Number of selected paths	Path distribution based on general evaluation			
			A	B	C	D
Task 1	Expert A	2		2		
	Expert A (second time)	1	1			
	Expert B	7	4	1	2	
	Expert B (second time)	6	3	3		
	Expert C	8	1	5	2	
	Expert D	3	1	1		1
Task 2	Expert A	1	1			
	Expert B	6	5	1		
	Expert C	7	2	4	1	
	Expert D	5	3	1	1	
Task 3	Expert B	8	4	2	2	
	Expert C	4	2	2		
	Expert D	3	3			
Total		61	30	22	8	1


(N) Nodes and links included in the paths of anticipated maps
(M) Nodes and links included in the paths of generated and selected by the experts
50
50
150
N∩M

Each area of circle represents the numbers of nodes and links included in paths. Note, the number in the circles represent not the actual number but the rates between each paths.

Fig. 7. The rate of paths

4.3.3 Experimental Results and Discussion

As results of the evaluation experiment, the four experts generated 31 maps in total and selected 61 paths of interest from the maps. Table 3 shows the distribution of paths selected from the maps and evaluated by the subjects, classified on the basis of the general evaluation. According to the results, the number of paths classified in the higher two levels was 30 for A: Interesting and 22 for B: Ordinary but important, which totals 52, occupying 85% of all the paths evaluated. Thus, we can conclude that it is possible with the tool to generate maps or paths sufficiently meaningful for experts. The extremely small number of D: Obviously wrong is attributable to the removal of unnecessary paths before the evaluation. However, this is conceivably not so problematic since the ratio of the unnecessary paths was about 70% to 80% of the paths existing just after map generation, and the paths can be removed by a simple operation[2]. While this result may seem to be rough and subjective, note here that our research goal is not formal analysis of ontologies from the point of views of ontology engineers and computer scientists but supporting domain experts to understand contents of ontologies. Different from formal analysis, content understanding is

[1] The numbers of sub concepts of *Environmental destruction problems* and *Tradeoff problems* are 43 and16 respectively. And the subjects can chose two kinds of focal points and two kinds of exploration methods.

[2] For example, when a node close to the center of a map is selected and paths are removed, all subpaths of the paths are removed, so that about 10% to 50% of all the paths are removed.

essentially subjective task. Thus, the evaluation method cannot avoid being subjective to some extent. Although subjective evaluations would be understood to be ad hoc and incomplete, the very fact that domain experts evaluate it as good/useful, which is subjective, is meaningful to them, and hence to our research. Therefore, we believe our evaluation result should not be taken as meaningless but taken as demonstrating some positive evaluation that suggests the benefit of our approach.

Then, using the maps generated by reproducing the originating scenarios for ontology enrichment by the detailed exploration described in Section 4.2.2 as "anticipated maps," we quantitatively compared the anticipated maps with the maps generated by the subjects (after removing unnecessary paths) in the experiment to evaluate whether maps having meaningful contents other than the scenarios anticipated at the time of ontology construction were generated through the experiment. We compared them through some calculations using the numbers of concepts and relationships which are included in paths of the anticipated maps, the maps generated and selected by subjects and both of them. They are shown as N, M and N∩M respectively in Fig.7. The results of the evaluation were as follows. First, of the paths included in the anticipated maps (it is calculated by "(N∩M) / N" in Fig.7), about 50% of them were included in the maps generated by the experts. This ratio indicates the ratio of problem structures that matched the paths of interests of experts in different domains among the problem structures anticipated as typical scenarios. Although the task settings in this experiment were not intended to reproduce the anticipated maps, conceivably, overlapping interests in the wide domain of biofuels appeared as overlapping interests among experts in different domains. On the other hand, the ratio of paths not included in the anticipated maps among the paths generated and selected by the experts, i.e., the ratio of paths not anticipated from the typical scenarios (it is calculated by "(M - N∩M) / M" in Fig.7), was about 75%. We think the result is enough to show that three quarters are new paths and the other quarter is included in the anticipated paths. Although it is difficult to objectively claim what is the best rate, it is meaningful enough to claim a positive support for the developed tool. This suggests that the tool has a sufficient possibility of presenting unexpected contents and stimulating conception by the user. In other words, this suggests that, by organizing the contents of the scenarios as generalized concepts in the ontology instead of directly storing the contents in a computer, more meaningful conceptual maps were generated compared with the case where the contents were stored intentionally in a computer.

Furthermore, the experts who served as subjects left positive comments about the maps they generated, such as: "Biomass resource production obviously involves human labor. However, we tend to focus on the material flow and environmental load control (restriction of greenhouse gas emissions, prevention of water pollution, and suppression of soil degradation) and tend to forget about the presence of human beings who support it." or "Normally, I wouldn't have noticed the path to sea pollution," indicating the possibility that the tool can contribute to overviewing problems or stimulating conception assistance. Also, in discussions after the experiment, we received one opinion: "It is useful that relationships in various domains are expressed in maps based on a single ontology." That is, we confirmed that the ontology exploration could serve to clarify relationships of knowledge, which tended to be segmented, and contribute to assisting in an interdisciplinary integrated understanding of the entire field.

5 Related Works

The novelty of our study lies in neither ontology visualization nor ontology navigation. The feature of our approach is an ontology exploration according to the users' viewpoints, for supporting domain experts to understand ontologies. We do not suppose to argue for novelty of the techniques which are used for visualization of the result of ontology exploration. While many researchers discuss techniques for ontology exploration and visualization since the early 2000s [7], they focus on formal aspects of ontology from the point of views of ontology engineers. Our idea is application of the techniques to bridge gaps between ontology and domain experts. Originally, our ontology editor (Hozo) have had sophisticated user interfaces for ontology visualization and navigation. It also got favorable comments such as from users in a workshop for comparing several ontology development tools. [8] mentions the comment as "it's not surprising that users liked the visual aids that some of the tools provided, such as Hozo's interactive graphs". However, in one of our research projects, such an interface has been proven to be insufficient for domain experts of sustainability science to understand the content of the ontology which is one of the significant goals. In other words, all the existing visualization tools assuming to allow users to see the entire structure of the ontology are not appropriate for domain experts to understand ontology. We therefore investigated the ontology exploration tool through this experience, and then it was well received by the domain experts. While it might be possible to generate paths using existing ontology navigation tools by choosing ways to trace in details, we suppose it is difficult to stimulate domain expert's way of thinking through divergently explorations because most of them are not designed for domain experts. While our tool uses concept maps as a format for visualization because it is familiar to domain experts, we have no intention to claim a novelty of the visualization method. The purpose of our tool is not concept map exploration but ontology exploration for understanding the ontology. A feature of our tool is that concept maps are generated from an ontology which systematizes domain knowledge as general as possible. It enables the users generate maps which includes contents anticipated by the developer of ontology.

Some researchers have developed tools for ontology exploration. TGVizTab (TouchGraph Visualization Tab) also supports a visualization which is similar to our tool. It aims for enhancement of clarification, verification and analysis of an ontology [9]. On the other hand, the features of our tool are functions for ontology exploration such as control of range of visualization, highlighting of their intensions and so on according to their viewpoints because we focus not on visualization but on exploration according to the users' viewpoints. Bosca develops a Protégé plug-in, called Ontosphere3D, which displays an ontology on a 3D space [10]. It allows the user to choose three kinds of visualization methods according his/her propose. The feature of this tool is a user definable set of ontology entities (concepts and relations) called Logical View. It is used for forming a hyper-surface in the multi-dimensional ontology space according to user's intensions. Although this approach to manage viewpoints is similar to our tool, it is different from the purpose of our tool that Ontosphere3D aims to provide ontology designers with a flexible instrument for effective representation and modeling of an ontology. Noppens proposes rendering techniques which combines visual analytics with interactive exploration for large scale structured data on the web [11]. When users are exploring an ontology, this tool clusters related entities according to their class hierarchies and inheritance information and visualize the result as clusters. It supports to discover hidden connections between individuals. While this tool focuses on exploration for large ontology with its instances (individuals), our tool aims at exploration for an ontology without instances. Helim also proposes an approach for interactive discovery relationships via the Semantic Web [12]. Tane discusses query-based multicontext theory for browsing ontologies [13]. In his approach, views for navigation are defined by sets of queries to knowledge bases. Though the search path function of our tool shares the same idea with them, it uses only simple combinations of relationships in an ontology because we focus on understanding the ontology. Christopher et.al discusses a constructive exploration of an ontology using world view and perspective [15]. Their work focuses on creating tight domain hierarchies from Folksonomies such as Wikipedia using mining techniques with some viewpoints. Exploring the knowledge space in Wikipedia is discussed as one of the techniques for mining. On the other hand, our approach focuses on understanding content of an ontology according to view points of domain experts.

6 Concluding Remarks and Future Work

This paper proposed divergent exploration to bridge a gap between an ontology which systematizes knowledge in generalized formats and domain experts who tend to understand the knowledge from domain-specific viewpoint. And we developed an ontology exploration tool for supporting users to explore an ontology divergently according to their intensions and viewpoints. This tool was applied to knowledge structuring of sustainability science. Through the experience, the tool was well received by domain experts in the domain and got favorable comment that it could contribute to proper structuring of knowledge across multiple domains. Then, we evaluated the tool through an experiment in cooperation with domain experts who are unfamiliar with ontology. As the result, we could make sure that domain experts could obtain meaningful knowledge for themselves as conceptual chains through the divergent exploration of ontology using the tool. The conceptual chains generated in the experiment included about 75% paths which were not supposed when the ontology was constructed. That is, we can say that the tool stimulated their way of thinking and contributed to obtaining unexpected conceptual chains which they have never thought. It is considered that the result is caused our approach for knowledge systematization that we should describe knowledge not as simple instances but as generalized concepts and relationships in an ontology. Furthermore, a domain expert who joined the experiment proposed us that our tool could be applicable for consensus-building among stakeholders in a workshop. It suggests that the tool clearly represents differences of thinking among stakeholders as differences of their viewpoints and conceptual chains generated by each of them so that they could understand ways of thinking of each other.

Future plan includes improvement of our tool to support more advanced problems such as consensus-building, policy-making and so on. We believe divergent exploration from multiple viewpoints contributes to making clear differences among standpoint, way of thinking and so on. For this purpose, we suppose to consider more intuitive user interfaces and visualizations for comparison between viewpoints. We also plan to be apply the ontology exploration tool for ontology refinement. In practice, we could use the tool to confirm whether the Biofuel ontology captures scenarios which were given by a domain expert and found some error in the ontology. We suppose it could be used for ontology refinement tool by domain experts to find not syntactic errors but content level faults in an ontology. We are trying to adopt this approach in a medical domain [15]. We think an evaluation of our tool on other ontologies is also important. Especially a large and complex ontology in OWL is a main target of the evaluation. While the system supports exploration of OWL ontologies, it does not support complex axioms in OWL. We will soon publish a new version of ontology explanation tool using OWL API with a reasoner.

Acknowledgements

This work was supported by Grant-in-Aid for Young Scientists (A) 20680009, the Global Environment Research Fund (Hc-082) of the Ministry of the Environment, Japan.

References

1. Gruber, T.: A translation approach to portable ontology specifications. In: Proc. of JKAW 1992, pp. 89–108 (1992)
2. Smith, B., et al.: The OBO Foundry: coordinated evolution of ontologies to support biomedical data integration. Nature Biotechnology 25(11), 1251–1255 (2007)
3. Kozaki, K., et al.: Hozo: An Environment for Building/Using Ontologies Based on a Fundamental Consideration of "Role" and "Relationship". In: Gómez-Pérez, A., Benjamins, V.R. (eds.) EKAW 2002. LNCS (LNAI), vol. 2473, pp. 213–218. Springer, Heidelberg (2002)
4. Mizoguchi, R., et al.: A Model of Roles within an Ontology Development Tool: Hozo. J. of Applied Ontology 2(2), 159–179 (2007)
5. Kozaki, K., et al.: Role Representation Model Using OWL and SWRL. In: Proc. of 2nd Workshop on Roles and Relationships in Object Oriented Programming, Multiagent Systems, and Ontologies, Berlin, July 30-31 (2007)
6. Kumazawa, T., et al.: Toward Knowledge Structuring of Sustainability Science Based on Ontology Engineering. Sustainability Science 4(1), 99–116 (2009)
7. Katifori, A., Halatsis, C., et al.: Ontology visualization methods - a survey. ACM Computing Surveys (CSUR) 39(4) (2007)
8. Noy, N.F., et al.: The CKC Challenge: Exploring Tools for Collaborative Knowledge Construction. IEEE Intelligent Systems 23(1), 64–68 (2008)

9. Alani, H.: TGVizTab: An ontology visualization extension for Protégé. In: Proc. of Knowledge Capture (K-Cap 2003), Workshop on Visualization Information in Knowledge Engineering, Sanibel Island, Florida (2003)
10. Bosca, A., Bonino, D.: Ontology Exploration through Logical Views in Protégé. In: Wagner, R., Revell, N., Pernul, G. (eds.) DEXA 2007. LNCS, vol. 4653, pp. 465–469. Springer, Heidelberg (2007)
11. Noppens, O., Liebig, T.: Understanding Interlinked Data - Visualising, Exploring, and Analysing Ontologies. In: Proc. of International Conferences on Knowledge Management and New Media Technology (I-KNOW 2008), Graz, Austria, pp. 341–348 (2008)
12. Heim, P., Lohmann, S., Stegemann, T.: Interactive Relationship Discovery via the Semantic Web. In: Aroyo, L., Antoniou, G., Hyvönen, E., ten Teije, A., Stuckenschmidt, H., Cabral, L., Tudorache, T. (eds.) ESWC 2010. LNCS, vol. 6088, pp. 303–317. Springer, Heidelberg (2010)
13. Thomas, C., et al.: Growing Fields of Interest - Using an Expand and Reduce Strategy for Domain Model Extraction. In: IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology 2008, vol. 1, pp. 496–502 (2008)
14. Tane, J., Cimiano, P., Hitzler, P.: Query-based multicontexts for knowledge base browsing: An evaluation. In: Schärfe, H., Hitzler, P., Øhrstrøm, P. (eds.) ICCS 2006. LNCS (LNAI), vol. 4068, pp. 413–426. Springer, Heidelberg (2006)
15. Ohta, M., Kozaki, K., Mizoguchi, R.: A quality assurance framework for ontology construction and refinement. In: Mugellini, E., Szczepaniak, P.S., Pettenati, M.C., Sokhn, M. (eds.) AWIC 2011. Advances in Intelligent and Soft Computing, vol. 86, pp. 207–216. Springer, Heidelberg (2011)

The Use of Foundational Ontologies in Ontology Development: An Empirical Assessment

C. Maria Keet

KRDB Research Centre, Free University of Bozen-Bolzano, Italy
School of Computer Science, University of KwaZulu-Natal, South Africa
keet@ukzn.ac.za

Abstract. There is an assumption that ontology developers will use a top-down approach by using a foundational ontology, because it purportedly speeds up ontology development and improves quality and interoperability of the domain ontology. Informal assessment of these assumptions reveals ambiguous results that are not only open to different interpretations but also such that foundational ontology usage is not foreseen in most methodologies. Therefore, we investigated these assumptions in a controlled experiment. After a lecture about DOLCE, BFO, and part-whole relations, one-third chose to start domain ontology development with an OWLized foundational ontology. On average, those who commenced with a foundational ontology added more new classes and class axioms, and significantly less object properties than those who started from scratch. No ontology contained errors regarding part-of vs. is-a. The comprehensive results show that the 'cost' incurred spending time getting acquainted with a foundational ontology compared to starting from scratch was more than made up for in size, understandability, and interoperability already within the limited time frame of the experiment.

1 Introduction

Ontologists tend to be outspoken about the usefulness of foundational (top-level) ontologies, such as BFO, DOLCE [1], GFO [2], and SUMO: either they are perceived to be essential or an impractical burden. Older ontology development methodologies that are still in use, such as METHONTOLOGY [3] and On-To-Knowledge [4], do not mention the use of a foundational ontology, but at the time of their development there were hardly any available, and the larger projects, such as GALEN and Cyc, developed their own. More recent methodologies, such as the NeON Methodology [5], mention it either in passing as part of ontology reuse in general, or explicitly, as in OntoSpec [6], and it is considered as an essential component in the OBO Foundry project [7,8]. A foundational ontology is also used with, among others, top-level domain ontologies, such as BioTop [9], in the coordination of lightweight ontologies and thesauri in KOKO [10], and with domain ontologies, e.g., [11,12]. The underlying assumptions of the proponents of the use of a foundational ontology are that ontology developers,

© Springer-Verlag Berlin Heidelberg 2011

once introduced to foundational ontologies, naturally will use it, be it right from the start or to align their ontology with one of them, because

i. it facilitates ontology development because one does not have to reinvent the wheel concerning basic categories and relations, and
ii. using a foundational ontology improves overall quality and interoperability.

On the other hand, such foundational ontologies are criticised as being too abstract, too expressive, too comprehensive for 'simple' or domain ontologies, and it takes too much time to understand them in sufficient detail. In addition, expressivity issues and the difference between a foundational ontology's take on how to represent attributes (e.g.,with qualities and qualia), and OWL's data properties with 'application ontologies' (*de facto*, OWLized formal conceptual data models) for ontology-driven information systems increases the perceived gap further [13]. Trying to answer whether the former or the latter stance holds cannot be carried out by simply collecting all ontologies on the web through, e.g., Swoogle or the TONES repository and counting the inclusion of a foundational ontology, because a substantial amount of them are experimental or tutorial ontologies or OWLized non-ontological resources (e.g., thesauri), one cannot always assess their quality regarding correct representation of the subject domain, the choice why a foundational ontology was used or not is unknown, and it is unknown how much resources went into developing the ontologies; hence, with this much uncertain parameters, one cannot draw any conclusions.

What is needed to commence ascertaining the validity of one or the other stance, are controlled experiments. This paper reports on one such experiment, which, to the best of our knowledge, is the first of its kind. After basic training in OWL and foundational ontologies (BFO, DOLCE, and part-whole relations), 52 participants developed 18 domain ontologies—all of them a "computer ontology"—in the timeframe of 24 hours. One third actually used a foundational ontology, and its developers had added, on average, more classes and class axioms than those who developed the ontology from scratch (albeit not statistically significant), and had added significantly less new object properties thanks to reusing those provided by the foundational ontology. Hence, the foundational ontology facilitated domain ontology development at least to the point that even with a very short timeframe, the investment required for using a foundational ontology was already more than evened out. In addition, they had slightly less errors and were more interoperable regarding the usage of part-whole relations in particular, thereby improving overall quality and interoperability. These results justify extending ontology development methodologies, or developing a new one, with a foundational ontology, both regarding when to choose one and how and where it aids the actual modelling.

The remainder of this paper is structured as follows. After describing the materials and methods of the experiment in Section 2, we present the results in Section 3 and discuss them in Section 4. We consider different directions of extensions of methodologies in Section 5, and conclude in Section 6.

2 Materials and Methods

The methodology for the experiment was as follows.

1. Lecture on purpose and usefulness of using a foundational ontology and overview of its contents (3-4 hours);
2. Divide course participants into smaller groups of 1-4 participants;
3. Provide the participants with instructions, being:
   (a) Develop a domain ontology about computers (i.e., it should contain that what you expect to find in a 'computer ontology' if you would search for it online);
   (b) You have the following input options:
      i. *tabula rasa*, i.e., start from scratch with an empty OWL ontology and do not import anything;
      ii. Use an OWLized foundational ontology (options provided: DOLCE, BFO, GFO);
      iii. And/or use the OWLized taxonomy of part-whole relations;
   (c) Name your ontology with the names of the group participants;
   (d) Time to develop the computer ontology: 24h from start to handing it in;
   (e) The ontology will not be graded, but is part of an experiment that will be discussed after having handed in the ontology;
4. Evaluation:
   (a) Assessment of the OWL files on usage of foundational ontologies, ontology metrics (language used, classes and object properties added etc.), and errors made;
   (b) Open questions with the participants regarding motivations of (non-) usage and modelling issues.

The materials used for the experiment were OWLized foundational ontologies provided to the course participants, including the respective URIs, being `DLP3971.zip` (a set of DOLCE ontologies), `bfo-1.1.owl` (BFO), `gfo.owl` (GFO), and two versions of the taxonomy of part-whole relations with a simplified DOLCE taxonomy, which is based on [14] (`pwrelations.owl` and an extended version `mereotopoDOLCE.owl`). A highly simplified version of DOLCE (^{my}DOLCE-lite$_{mini}$, for short) is depicted in Fig. 1 and a summary of the OWLized taxonomy of part-whole relations is included in Fig. 2. The ontology development environment was Protégé 4.1beta with the integrated Hermit v1.2.4 automated reasoner. No restrictions were put on the participants to use, or not, non-ontological resources, such as textbooks, Wikipedia, product catalogs etc.

3 Results

In this section we describe the setting of the experiment, provide a characterisation of the participants, and describe quantitative and qualitative results of the ontologies that were developed. This will be discussed afterward in Section 4.

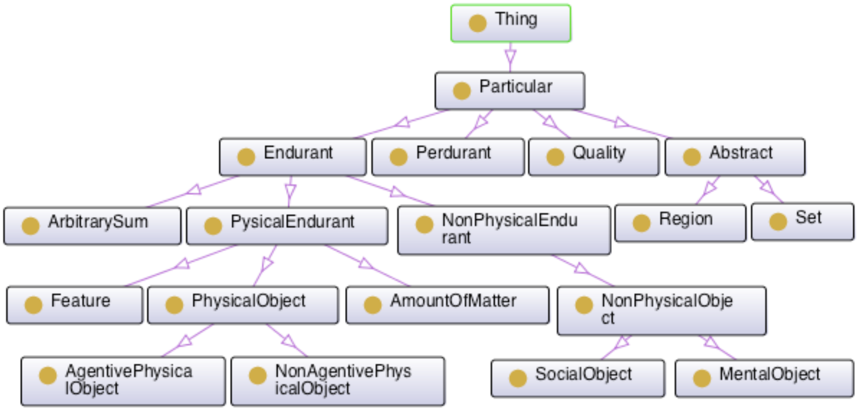


Fig. 1. Graphical rendering of ^{my}DOLCE-lite$_{mini}$ in the Protégé 4.1 OntoGraf tab

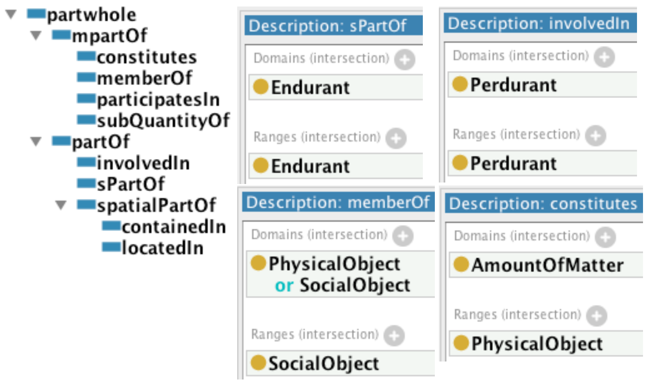


Fig. 2. The OWLized version of the taxonomy of part-whole relations [14] (left) with screenshots of domain and range restrictions of four object properties (right)

3.1 Setting

The experiment was carried out in three sessions during a course on a comprehensive introduction to ontology engineering at Universidad de la Habana (UH) and Universidad de Ciencias Informáticas (UCI) in Cuba, and CSIR Meraka in South Africa in 2010. Regarding the course outline and content, for indicative purpose, the syllabus and slides of the latest installment are available at `http://www.meteck.org/teaching/SA/MOWS10OntoEngCouse.html`. After a session on OWL and OWL 2, they contained about a 1.5 hour introduction in foundational ontologies in general and DOLCE and BFO in particular, which was followed by 1-1.5 hours on part-whole relations. The experiment commenced afterward in the labs.

The reason for choosing the subject domain of computers for the experiment is that most participants can be considered domain experts in that area (see

Section 3.2) and it lends itself well not only for a wide range of different entity types but also the need to use part-whole relations in one way or another. In addition, at the time of the experiment, it was asserted that there was no computer ontology or similar available online and therefore no bother to search for it. (In fact, there is a serious ontology about software and programming languages [12], but it was inaccessible to the participants at the time of the experiment.)

3.2 Characterisation of the Participants

The amount of participants in the courses who handed in ontologies is 16 (UH) + 27 (UCI) + 8 (Meraka) = 52. One of the participants was a biologist with an interest in ontologies, three were in interdisciplinary areas (juridical AI, IT & education, and computational linguistics) and the remaining 48 participants were computer scientist; hence, 51 participants did have at least some modelling experience, have had at least one logic course, and can be considered also domain experts. No participant did have any formal training on OWL before the course and only two participants in the last session had had a course on Description Logics. The participants of the last session did receive some training on Protégé prior to the ontology engineering course, whereas for the other two installments, the Pizza Ontology tutorial was advised as self-study and several exercises were carried out in the preceding days. Thus, all participants were relatively novice ontology developers.

Most of the participants were studying either for a MSc or PhD (n=48), or a researcher, lecturer or professor (n=4); lecturers who are also MSc or PhD student—a considerable amount—are counted in the former group. The participants' age was predominantly between 23-33 years, with five in the 45-65 year age bracket. 19 participants were female and 33 male. The subgroups the participants formed themselves were mixed.

The participants were principally interested in ontology engineering in that they needed to develop a domain ontology for their research projects, i.e., as a component of an ontology-driven information system, whereas a small amount (about 5) were carrying out research for ontologies, such as debugging ontologies and user interfaces.

3.3 Assessment of the Ontologies

Ontology data and statistics. The 52 participants developed 6 (UH) + 8 (UCI) + 4 (Meraka) = 18 ontologies in groups of 1-5 participants. Six groups, or 1/3, used a foundational ontology, of which one imported the full DOLCE (all ontologies in `DLP3971.zip`), one DOLCE-Lite, two used `pwrelations.owl` that has both a taxonomy of part-whole relations and the DOLCE taxonomy of categories, and two groups used its extended version `mereotopoDOLCE.owl`; hence, no group used BFO or GFO.

Table 1 contains basic data of the slightly anonymized ontologies. In the remainder of the analysis, we exclude the biologist outlier (`52`) to ensure homogeneity in the notion of type of participant (this participant did not know what to add and was unfamiliar with logic [pers. comm.]).

Table 1. Basic characteristics of the computer ontologies developed by the participants; for each participant in a subgroup, a two-digit number was used in order of handing in the ontology. The language (DL) fragment was obtained from the Protégé 4.1beta 'active ontology' tab and is for indicative purpose only. 'Found. onto.' = usage of a foundational ontology.

Parameter ⇒ / **Ontology** ⇓	From scratch	Found. onto. DOL-CE	Found. onto. pwrel/ mereo-topo	Language (DL fragment)	New entities class	New entities obj prop	New entities data prop	New entities indi-vi-duals	new class axioms
010203.owl	+			ALCIQ	16	4	0	9	34
0405.owl	+			SROQ	12	3	0	26	22
06070809.owl	+			ALCIQ	17	3	0	28	30
10.owl	+			ALCHI	17	6	0	1	18
111213.owl			+	SRIQ(D)	16	5	2	9	18
141516.owl			+	SRIQ	39	3	0	6	50
171819.owl	+			ALCHIF	20	6	0	2	20
202122.owl		+		SHIQ	44	2	0	0	52
232425.owl	+			ALCHIQ(D)	24	5	0	0	54
26272829.owl	+			ALCHIQ(D)	25	6	4	1	40
30.owl	+			ALCQ	36	2	0	0	44
3132333435.owl	+			ALCHF	36	4	0	0	55
36373839.owl			+	ALCROIQ(D)	24	0	7	3	34
40414243.owl			+	ALCROIQ(D)	22	0	4	3	32
444546.owl	+			AL	27	1	0	0	24
4748.owl	+			SHI	13	4	0	0	16
495051.owl	+			ALCI	10	5	1	11	13
52.owl		+		SHOIN(D)	1	0	0	0	1

In addition to the strict division between starting from scratch and using a foundational ontology, and for the purpose of analysis, we also consider a group of ontologies where its developers did inspect a foundational ontology, but did not use one in the submitted OWL file, being, at least, 0405, 10, 30, 3132333435, and 4748. Upon inquiry, the main reason for not using one after all was time constraints and 4748 used the so-called Componency Ontology Design Pattern because either one of the foundational ontologies was "too much to handle" yet there was still the desire to reuse some existing material. Assessing averages, median, and standard deviation (Table 2), they are very similar to those who started from scratch, in particular compared to the substantial differences with those who started with a foundational ontology, and therefore this 'would have but did not do'-subgroup is not considered further as a specific subgroup.

While the differences in average and median are particularly favourable for those ontologies who started with a foundational ontology—i.e., having more classes and class axioms despite losing time in editing due to getting acquainted with the foundational ontologies—one also can observe quite some variation among the individual ontologies in Table 1. To this end, a Student t-test was performed on the two groups regarding new classes, new class axioms, and new

Table 2. Basic analysis of the new additions to the submitted ontologies; numbers are rounded off

Parameter ⇒ **Group ⇓**		New entities class	obj. prop.	data prop.	individuals	New class axioms
All	Average	23.4	3.5	1.1	5.8	32.7
	Median	22	4	0	2	32
	StDev	10.1	2.0	2.0	8.8	14.3
Found. onto. reuse	Average	29	2	2,6	4.2	37.2
	Median	24	2	2	3	34
	StDev	11.9	2.1	3.0	3.4	14.0
From scratch	Average	21.1	4.1	0.4	6.5	30.8
	Median	18.5	4	0	1	27
	StDev	8.7	1.6	1.2	10.3	14.6
Inspect found. onto.	Average	22.8	3.8	0	5.4	31
	Median	17	4	0	0	22
	StDev	12.2	1.5	0	11.5	17.5

object properties. p=0.145 for new classes, hence, barely not significant to claim starting with a foundational ontology significantly speeds up ontology development. For new class axioms, p=0.420, hence, one cannot conclude anything either way. For new object properties, however, p=0.043, or: those who started with a foundational ontology added significantly less properties than those who started from scratch. Of the groups who started from scratch, 10 out of 12 invented a part-whole object property of their own, having names such as `hasPart`, `esParteDe` ('is part of' in Spanish), `compuestaPor` ('composed of' in Spanish), `hasComponent`, and so forth. Conversely, the groups who reused a foundational ontology availed of those part-whole object properties already present in the imported ontology—which, consequently, have a clear meaning compared to those in the other 10 ontologies.

A noteworthy observation is that, in analogy with software development, ontology development is not a factory line—more people in a group did not result in larger ontologies in the same amount of time. Other observations are that about 2/3 of the groups used qualified number restrictions and therewith uses OWL 2 DL compared to one of its profiles, and the ontologies in the most expressive OWL 2 DL fragment are typically those that reuse a foundational ontology. The data also shows that more emphasis had been put on adding a class hierarchy than class axioms involving object properties.

Qualitative aspects. In addition to the basic characteristics, one has to consider the quality of the ontologies concerning both the contents and the modelling errors. Considering the errors, the following can be observed. Unlike the well-known common error of confusing part-of with is-a among novice modellers, *none* of the 18 ontologies had this error. There were multiple cases of is-a vs instance-of confusion where types of processors and motherboards were modelled as instances, and a few 'unexpected' results were encountered with the reasoner during development due to domain and range restrictions that were too restrictive in hindsight or had an axiom instead of the intended atomic class. There were several ontologies where the "NonSimpleRoleInNumberRestriction" was encountered either already during the development or after handing in the ontology, which was due to the use of `Min-`, `Max-`, or `ExactCardinality` with a non-simple object property. This was due to the interaction with the characteristics of the part-whole property that was not a simple object property anymore (see [15], of which the technical details about constraints on roles are described in [16]). A relatively minor issue concerns the difference between the naming of the ontology, i.e., changing the URI from a default like `.../Ontology123456789.owl` into a meaningful name, versus naming the OWL file: the participants did the latter, but not the former. Thus, while the sensitization of part-whole relations prevented one type of common errors, these errors observed are general mistakes that are not attributable to not using a foundational ontology.

The six ontologies that used a foundational ontology were analysed further. `141516` has `PC` as a subclass of DOLCE's `AgentivePhysicalObject` (APO), `40414243` has `Ordenador` (computer) as a subclass of `ArbitrarySum` (AS), and the other four have it as a subclass of `NonAgentivePhysicalObject` (NAPO). `40414243` motivated that each computer is a "varying collection of things", and "therefore" an arbitrary sum, whereas `141516` deemed APO appropriate because a computer "is a physical object that does things". DOLCE's motivation to distinguish between APO and NAPO, however, is that the former is assumed to have beliefs, desires, and intentions and are typically attributed only to persons, and the 'arbitrary' in AS is to be taken more arbitrarily than those collections that make up a computer (for which one can identify constraints) [1]; hence, computer as an (in-)direct subclass of NAPO is the appropriate category. A straightforward explanation of the details of these DOLCE categories immediately resolved the difference, converging to NAPO. Eleven of the 12 other ontologies had added `Computer` (or similar) to the ontology, but none was the same or even alike and a resolution was not attempted due to time constraints.

4 Discussion

We discuss the test results and limitations of the set up in order with the claims regarding the benefits and problems of using a foundational ontology, and subsequently consider other factors that did or might affect the results.

4.1 Reuse of Entities vs. Too Comprehensive and Too Complicated

The main explanation for why the groups who used a foundational ontology did not lag behind those who started from scratch—in fact, quite to the contrary—is that they availed of the imported classes and object properties, so there was no time lost with, among others, discussing how the part-whole relation should be named, thereby avoiding having to reinvent the wheel concerning basic classes and object properties. From the other perspective, one might assume that choosing the right category from a foundational ontology is time-consuming. If the latter were the case, and provided one wants to import a foundational ontology, then it certainly would have been easier for the former group to import BFO instead of DOLCE or the taxonomy of part-whole relations, because BFO is a bare taxonomy of 39 classes, 0 object properties, and 107 class axioms (in the DL language $\mathcal{ALC}$)[1] compared to 37 classes, 70 object properties, and 94 class axioms in DOLCE-Lite (in $\mathcal{SHI}$; the much larger OLWized DOLCE is in $\mathcal{SHOIN}(\mathcal{D})$) and the part-whole relations with the ^{my}DOLCE-lite$_{mini}$ has 18 classes, 13 object properties, and 31 class axioms (in $\mathcal{SRI}$). Yet, this did not occur and therefore this informal criticism cannot be substantiated with the data obtained in the experiment, whereas the former claim of speeding up ontology development by using a foundational ontology, can. Nevertheless, in this context it is worthwhile to observe that the current OWLized DOLCE versions are too expressive for the OWL 2 EL and OWL 2 QL profiles [17], which may affect its use and reuse. In addition, the developers of 36373839 and 40414243 *manually deleted* classes and object properties because, according to its developers, they were perceived to be unnecessary and cluttering the ontology. This indicates a possible use for partial imports that is currently still not implemented (works in ontology modules is in progress [18]), or a 'hide' feature either in the graphical interface or also at the logical level, as implemented in, e.g., CASL [19]. A follow-up experiment may want to include a scenario with more different—expressive and slimmed—versions of DOLCE to figure out what is, or are, the 'optimal' DOLCE version(s) for practical ontology engineering.

It is unclear why the participants chose DOLCE over BFO; that is, despite asking for it, the answers were not of sufficient detail to warrant drawing any conclusions. If we assume some foundational ontology FO_A is more suitable for tasks T_A or subject domain D_A and FO_B for T_B or D_B, then this should be known and have objective arguments why this is the case so that it can be taken into account in ontology development. We are not aware of the existence of such an assessment, so that it is more likely that developers—be it in this experiment or for real ontology development—choose a particular foundational ontology for compatibility with other existing or envisioned ontologies and/or infrastructure, or philosophical motivations, or subjective preferences.

4.2 Quality and Interoperability

A general, and well-known, problem in assessing the second claim mentioned in the introduction—a better quality ontology—is to determine what are the unambiguous objective parameters by which one can say which ontology is really a *better* ontology. We took only a minimal approach to it in the assessment, such as the actual errors made (is-a vs instance-of) and avoided (is-a vs part-of), the ambiguity of names/labels that especially in the 'from scratch' ontologies

[1] It is claimed recently [8] that "BFO" now has to be understood as BFO + the Relation Ontology; the RO passed the revue in the lectures, but because of the ambiguous status of the combination at the time, it was not included in the experiment.

bore little semantics, if any (compuestaPor), logically correct but unintended mistakes with respect to the subject domain, and language errors (non-simple role in number restriction). Based on such basic metrics, the subgroups who used a foundational ontology fared a little better, but it is not entirely free of debate.

In addition, it may neither prevent nor fully solve certain differences in representation of knowledge. For instance, the ontologies that reused a foundational ontology had `Computer` at three different places in the DOLCE taxonomy, and, overall, 17 ontologies had `Computer` (or similar or a synonym) added as a class to the ontology, with 5 of them as a defined concept, and none was the same. There were three different principle directions taken by the subgroups: either computer was some collection of macro-components, such as tower, keyboard, and monitor, or it was some collection of its parts, such as RAM, CPU, motherboard etc., or something that has both hardware and software, where two ontologies had two of the three perspectives combined (`232425` and `26272829`). This surprised the participants, especially because intuitively it is obvious what a computer is and they were of the opinion that they were describing the characteristics of the same physical objects. How this has to be resolved in the realist-BFO way [8] or some other, perhaps more practical, way [20], is a different topic.

Compared to `Computer`, this was easier for `Software`, partially thanks to its underspecification in the ontologies and partially thanks to the fact that there is a domain ontology about software and programs that extends DOLCE [12]. As it turns out, it is easy to align at least the participants' DOLCE-based computer ontologies with this one. More precisely, four of the six ontologies had `Software` in their ontology, of which one as a subclass of AgentivePhysicalObject (like it did with `Computer` and which can simply be resolved in the same manner), and three had it as a subclass of NonPhysicalObject (`202122`, `36373839`, and `40414243`); that is, in the same branch as the more refined ontology by Lando and co-authors [12] and therewith relatively easy to merge.

4.3 Other Factors

A complicating factor in ontology development in the first two sessions of the experiment was the natural language barrier in conjunction with the new design of Protégé 4. Whereas Protégé 3 uses icons for the familiar Description Logic symbols ($\forall$, $\sqcap$ etc), they have been changed into keywords in Protégé 4; more precisely, *English* keywords. Such an 'anglification' is not helpful in an international setting, which is a setting that ought to have been assumed for tools for the Semantic Web as part of its internationalization objective [21]. It was not intuitive to figure out what the keywords were (compared to immediate understanding of the symbols) so that modellers lost time finding the appropriate ones and it resulted in ugly spanglish that hampered understandability, such as `Ordenador subClassOf utiliza exactly 2 Perifericos_Principales` (vs. the more intelligible `Ordenador` $\sqsubseteq$ $=$ 2 `utiliza.Perifericos_Principales` that would have been obtained with Protégé 3). For proper internationalization and supporting ontology development environments tailored to subject domain experts, the keywords should be provided in various languages (or to provide the option to add them for one's preferred language), and in order to cater for different types of modellers, an option to switch from keywords back to the natural language-independent symbols will be welcome.

Concerning the preparatory training of the participants, it may seem possible to argue in three directions regarding foundational ontology reuse: either 1/3 is a low reuse percentage because the lecturer had not taught the matter sufficiently well, or that even with good teaching there is just 1/3 of the groups who reused a foundational ontology voluntarily, or that thanks to good teaching it is an impressive 1/3 of the ontologies where people voluntarily reused a foundational ontology. Aside from the formal and informal student evaluations (unanimously positive for the first and third installment), this can be settled partially by carrying out the experiment with other lecturers, but this is beyond the scope of the current experiment. In addition, the purpose of the experiment was not communicated to the participants, because one of the parameters was to examine how many groups would *voluntarily* choose to use a foundational ontology without suggestive interference. The downside of this was that groups followed three distinct strategies in ontology development: either they were focussed on adding as much as possible ('adding more entities is better') or they were more concerned with discussions how to model the various entities as good as possible (e.g., "what constitutes a computer?", "is software is a physical object?"), or experimenting with the reasoner (e.g., "will `wrong_computer3` make the ontology inconsistent?"). Follow-up experiments may want to focus solely on the ontology quality dimension, informing the participants about this beforehand, and, by dividing the subgroups into two: one where the people are forced to use a foundational ontology, one where they should not.

Last, although the time allotted to domain ontology development may seem short, one has to bear in mind that the developers were also wearing their hat as domain experts and did receive logic training beforehand, thereby mitigating the short timeframe. While the participants in each installment of the course were highly motivated, this might be different for other experiments so that it may be beneficial to build in more precise timing with compulsory lab sessions.

5 On Enriching Methodologies

Given the cautiously positive outcome in favour of reuse of a foundational ontology, one has to look ahead at where, how, and in which methodology this can be incorporated, which requires inclusion of at least two main components: choosing which foundational ontology to reuse and how to use it in the modelling process.

5.1 Extending High-Level Methodologies to Include a Foundational Ontology Usage Step

Including a decision point to choose a foundational ontology somewhere in the procedure is fairly straightforward, be it by extending existing methodologies that do not address foundational ontologies explicitly yet or a new methodology based on a set of criteria the methodology has to meet (such as outlined in [22,5]). For instance, for the relatively well-known METHONTOLOGY, the addition would be both in the "conceptualization" stage that has intermediate representations made by the domain experts and in the "formalization" stage where the domain-expert understandable model is transformed into a formal or semi-computable model. For the former case, the ontology can be offered in any format because it is used for modelling guidance only; for the latter case, and assuming a Semantic Web setting, then the ontology should be available in one of the OWL species. For the NeON methodology [5], this means extending its "Scenario 3" with an explicit section on 'foundational ontology' (in addition to the current "general or common ontology") and creating a new so-called "Filling card" for foundational ontologies, which can have the following contents according to the standard filling card headings:

– *Definition*: Foundational Ontology Reuse refers to the process of using a foundational ontology to solve different problems, such as non-interoperability of ontologies and losing time reinventing known modelling solutions.
– *Goal*: The goal of this process is to find and select a foundational ontology that is either to be integrated in the stand-alone ontology or ontology network being developed or to be imported at the start of ontology development.
– *Input*: Competency questions included in the ontology requirements specification document of the ontology to be developed (see [5]), at least one file in an implementation language for each such ontology, and, when available, a (set of) table(s) comparing the candidate foundational ontologies to be reused across the same criteria.
– *Output*: A foundational ontology integrated in the ontology being developed.
– *Who*: Ontology developers involved in the ontology development, such as domain experts, knowledge engineers, and practice-oriented philosophers.
– *When*: The foundational ontology reuse process should be carried out after the "ontology specification activity" and before other "ontological resource reuse".

Such a high-level filling card, however, does not yet aid the modeller in updating the contents of the ontology, for which we have to look at the second type of methodologies in the next section.

5.2 Augmenting the Modelling Exercise

Neither one of the high-level extensions says anything about how to choose between one or the other foundational ontology or how the chosen ontology is to be integrated in the modelling exercise. Concerning the former, the lofty goal of an "ontology library" of interchangeable foundational ontologies that was envisioned in [1]—thereby avoiding the need to choose between one or the other foundational ontology—is yet to be realised. Concerning the latter, we have to look at another 'type' of methodology[2]. Noy and McGuinness' Ontology Development 101 (OD101) [23] and Kassel's OntoSpec [6] focus specifically on how to represent entities in an ontology, such as conducting a property analysis, assessing cardinalities, distinguishing between is-a or instance-of and so forth, whereby OntoSpec relies on the OntoClean and DOLCE foundations to stimulate good modelling practices and OD101 is based on the authors' own experiences in ontology development. To the best of our knowledge, there are no tutorial ontologies and exercises in conjunction with DOLCE, BFO or GFO[3] and practical examples may need to be added at least to the lectures but even more so in a structured fashion for all leaf categories of the foundational ontologies in each methodology. For OD101, this means rewriting steps 4-7 and sections 4-6 to reflect the guidance from foundational ontologies. For OntoSpec, it means extending the few elaborate examples to cover all DOLCE leaf types and, assuming it is to be used within a Semantic Web setting, converting its current representation from DOLCE-OS (a "semi-informal OntoSpec language" [6]) into a suitable OWL species. For both cases, however, a new solution to the issue of choosing the appropriate part-whole relation is required. This may be achieved by automating the decision diagram in [24] extended with a 'cheat sheet' with examples and informal definitions for each of the foundational ontology's main categories (alike Table 1 in [1]).

A brief illustration is described in the following example.

Example. Let us take the African Wildlife tutorial Ontology, and the wish to represent that the elephant's tusks (ivory) are made of apatite. There are three classes: `Elephant`, `Tusk`, and `Apatite`. Although reading the text in [1] would be better, in this case its examples in Table 1 are already helpful: examples for Non-Agentive Physical Object (NAPO) are "a hammer, a house, a computer, a human body" and for Amount of Matter (M) they are "some air, some gold, some cement"; hence, `Elephant` and `Tusk` are subclasses of NAPO, and `Apatite` is a subclass of M. Then, considering the relations between them, we can avail of the OWLized part-whole relations property hierarchy, which has its properties typed with the DOLCE categories (recollect Fig. 2). Thanks to knowing the DOLCE category of each of the three classes, it straightforwardly follows that each tusk is a structural part of, `sPartOf`, elephant in our wildlife ontology and that tusk is `constitutedOf` apatite. ♢

These suggestions for enhancing extant ontology development methodologies are incomplete. However, note that the aim of this work was first to examine whether it makes sense practically to use foundational ontologies, if they are used voluntarily, and how, in a mode that is based not only on theoretical motivations, but, moreover, whether this can be motivated from the perspective of hard data

[2] That is, at present there are two strands of methodologies, but they may well become integrated into one larger methodology at a later stage.

[3] except for informal notes with very few examples at `http://keet.wordpress.com/2010/08/20/african-wildlife-ontology-tutorial-ontologies/` by this author and at `http://www.itee.uq.edu.au/~infs3101/` by Robert M. Colomb.

in ontology development. As was demonstrated with the results obtained in the experiment, a comprehensive extension of the methodologies indeed will be of practical use.

6 Conclusions

We have investigated assumptions surrounding foundational ontology reuse in a controlled experiment with 52 developers who designed 18 domain ontologies. One-third of the ontologies were developed using a foundational ontology. Concerning the contents, on average, those who commenced with a foundational ontology added more classes, more class axioms, and significantly less object properties. The comprehensive results showed that the 'cost' incurred in spending time getting acquainted with a foundational ontology compared to starting from scratch was more than made up for in better quality and interoperability already with the limited duration of the experiment. Because of the positive results, we considered possible extensions to extant methodologies, which have to be at two levels: choosing which foundational ontology to reuse and how to use it in the modelling.

We are working on a tool to help choosing the appropriate part-whole relations and consider future extensions to help choosing when one foundational ontology would be better to reuse than another. It requires further investigation why the participants preferred DOLCE over BFO, and what the outcome will be if also much larger ontologies such as Cyc or SUMO were to be added to the options in a controlled experiment. It may be interesting to see similar experiments with other types of participants, such as with non-computing domain experts with experience in modelling.

Acknowledgements. The visit to CSIR Meraka, South Africa, was partially funded by the EU FP7 Net2 project "Enabling a network of Networked Knowledge" and partially by "Technologies for Conceptual Modelling and Intelligent Query Formulation" within the Executive Programme of Scientific and Technological Co-Operation between the Italian Republic and Republic of South Africa.

References

1. Masolo, C., Borgo, S., Gangemi, A., Guarino, N., Oltramari, A.: Ontology library. WonderWeb Deliverable D18, `http://wonderweb.semanticweb.org` (ver. 1.0, 31-12-2003)
2. Herre, H., Heller, B.: Semantic foundations of medical information systems based on top-level ontologies. Knowledge-Based Systems 19, 107–115 (2006)
3. Fernandez, M., Gomez-Perez, A., Pazos, A., Pazos, J.: Building a chemical ontology using METHONTOLOGY and the ontology design environment. IEEE Expert: Special Issue on Uses of Ontologies, 37–46 (January/February 1999)
4. Staab, S., Schnurr, H., Studer, R., Sure, Y.: Knowledge processes and ontologies. IEEE Intelligent Systems 16(1), 26–34 (2001)
5. Suarez-Figueroa, M.C., et al.: NeOn methodology for building contextualized ontology networks. NeOn Deliverable D5.4.1, NeOn Project (2008)

6. Kassel, G.: Integration of the DOLCE top-level ontology into the OntoSpec methodology. Technical Report hal-00012203, Laboratoire de Recherche en Informatique d'Amiens (October 2005), `http://hal.archives-ouvertes.fr/ccsd-00012203`
7. Smith, B., et al.: The OBO Foundry: Coordinated evolution of ontologies to support biomedical data integration. Nature Biotechnology 25(11), 1251–1255 (2007)
8. Smith, B., Ceusters, W.: Ontological realism: A methodology for coordinated evolution of scientific ontologies. Applied Ontology 5, 79–108 (2010)
9. Beisswanger, E., Schulz, S., Stenzhorn, H., Hahn, U.: BioTop: An upper domain ontology for the life sciences. Applied Ontology 3(4), 205–212 (2008)
10. Hyvönen, E.: Preventing ontology interoperability problems instead of solving them. Semantic Web 1, 33–37 (2010)
11. Keet, C.M.: Factors affecting ontology development in ecology. In: Ludäscher, B., Raschid, L. (eds.) DILS 2005. LNCS (LNBI), vol. 3615, pp. 46–62. Springer, Heidelberg (2005)
12. Lando, P., Lapujade, A., Kassel, G., Fürst, F.: Towards a general ontology of computer programs. In: Proceedings of the 2nd International Conference on Software and data Technologies: ICSOFT 2007, Barcelona, Spain, July 25-27 (2007)
13. Keet, C.M.: Dependencies between ontology design parameters. International Journal of Metadata, Semantics and Ontologies 5(4), 265–284 (2010)
14. Keet, C.M., Artale, A.: Representing and reasoning over a taxonomy of part-whole relations. Applied Ontology 3(1-2), 91–110 (2008)
15. Motik, B., Patel-Schneider, P.F., Parsia, B.: OWL 2 web ontology language structural specification and functional-style syntax. W3c recommendation, W3C (October 27, 2009), `http://www.w3.org/TR/owl2-syntax/`
16. Horrocks, I., Kutz, O., Sattler, U.: The even more irresistible $\mathcal{SROIQ}$. In: Proceedings of KR 2006, pp. 452–457 (2006)
17. Motik, B., Grau, B.C., Horrocks, I., Wu, Z., Fokoue, A., Lutz, C.: OWL 2 Web Ontology Language Profiles. W3c recommendation, W3C (October 27, 2009), `http://www.w3.org/TR/owl2-profiles/`
18. Stuckenschmidt, H., Parent, C., Spaccapietra, S. (eds.): Modular Ontologies—Concepts, Theories and Techniques for Knowledge Modularization. Springer, Heidelberg (2009)
19. Kutz, O., Lücke, D., Mossakowski, T., Normann, I.: The OWL in the CASL – designing ontologies across logics. In: Proc. of OWLED 2008, Karlsruhe, Germany, October 26-27 (2008)
20. Lord, P., Stevens, R.: Adding a little reality to building ontologies for biology. PLoS ONE 5(9), e12258 (2010)
21. Horrocks, I., Patel-Schneider, P.F., van Harmelen, F.: From SHIQ and RDF to OWL: The making of a web ontology language. J. Web Semantics 1(1), 7 (2003)
22. Dahlem, N., Guo, J., Hahn, A., Reinel, M.: Towards an user-friendly ontology design methodology. In: International Conference on Interoperability for Enterprise Software and Applications (I-ESA 2009), Beijing, China, April 21-22, pp. 180–186 (2009)
23. Noy, N., McGuinness, D.: Ontology development 101: A guide to creating your first ontology. Technical Report KSL-01-05, Stanford Knowledge Systems Laboratory (March 2001)
24. Keet, C.M.: Part-whole relations in object-role models. In: Meersman, R., Tari, Z., Herrero, P. (eds.) OTM 2006 Workshops. LNCS, vol. 4278, pp. 1118–1127. Springer, Heidelberg (2006)

Using Pseudo Feedback to Improve Cross-Lingual Ontology Mapping

Bo Fu, Rob Brennan, and Declan O'Sullivan

Knowledge and Data Engineering Group, School of Computer Science and Statistics,
Trinity College Dublin, Ireland
{bofu,rob.brennan,declan.osullivan}@cs.tcd.ie

Abstract. Translation techniques are often employed by cross-lingual ontology mapping (CLOM) approaches to turn a cross-lingual mapping problem into a monolingual mapping problem which can then be solved by state of the art monolingual ontology matching tools. However in the process of doing so, noisy translations can compromise the quality of the matches generated by the subsequent monolingual matching techniques. In this paper, a novel approach to improve the quality of cross-lingual ontology mapping is presented and evaluated. The proposed approach adopts the pseudo feedback technique that is similar to the well understood relevance feedback mechanism used in the field of information retrieval. It is shown through the evaluation that pseudo feedback can improve the matching quality in a CLOM scenario.

Keywords: Cross-Lingual Ontology Mapping; Pseudo Feedback.

1 Introduction

One approach to ontology construction is to use language neutral identifiers to label concepts [1], whereby ontological entities are natural language independent. Given such ontologies, there would be little need for cross-lingual ontology mapping. However, as Bateman points out "the path towards viable ontologies is one that is irreconcilably connected to natural language" [2]. With this view to ontology construction being largely adopted in practice [3], multilinguality is increasingly evident in ontologies as experts with various natural language preferences build knowledge representations in multilingual organisations [4], government regulations [5], medical practice [6], to just name a few. As a result, notable research can be seen in the area of multilingual ontology acquisition [7], linguistic enrichment of ontologies [8] and ontology localisation [9]. These efforts highlight the importance of dealing with multilingual ontologies, and the ability to reason over knowledge bases regardless of the natural languages in them has become a pressing issue in digital content management. Ontology mapping techniques must be able to work with otherwise isolated ontologies that are labelled in diverse natural languages.

One way to achieve semantic interoperability across natural language barriers is by means of cross-lingual ontology mapping (CLOM). A valid approach to CLOM is to translate the labels of a given ontology to the natural language used by the other ontology(ies) first, and apply monolingual ontology matching techniques next, as

© Springer-Verlag Berlin Heidelberg 2011

demonstrated in [10, 11, 12, 13]. A key challenge involved in this approach is ensuring the translated labels will maximise the final matching quality, since noisy translations could potentially pose negative impact on the monolingual matching tools as shown in [14]. Previous work [15] shows that selecting suitable translations is critical to the generation of quality CLOM results. Motivated by this requirement, this paper presents a novel approach that uses pseudo feedback, which is inspired by the well understood relevance feedback mechanism commonly used in information retrieval, to select ontology label translations as a way to improve CLOM. The proposed approach is evaluated against a baseline system in an experiment that uses the Ontology Alignment Evaluation Initiative (OAEI) 2009 benchmark dataset and the OAEI gold standard involving ontologies labelled in English and French. The evaluation results suggest that the pseudo feedback feature improves the CLOM quality comparing to the baseline system.

The remainder of this paper is organised as follows. Some related work is outlined in section 2. A pressing challenge for current CLOM approaches is discussed in section 3. To address this challenge, the pseudo feedback feature to improve CLOM is proposed in section 4. This proposed approach is evaluated in a CLOM experiment discussed in section 5. Finally, conclusions and future work are discussed in section 6.

2 Related Work

Current approaches to CLOM can be summarised as follows, manual processing [16], corpus-based [17], via linguistic enrichment [18], via indirect alignment [19] and translation-based [10, 11, 12]. An example of manual CLOM is discussed in [16], where an English thesaurus is mapped to a Chinese thesaurus by hand. Given large and complex ontologies, such a time-consuming and labour-intensive approach may be infeasible. Ngai et al. [17] use a bilingual corpus to align WordNet (in English) and HowNet (in Chinese), however, as such corpora are not always available to domain-specific ontologies, this approach may be unsuitable in some CLOM scenarios. Pazienza & Stellato [18] propose a linguistically motivated mapping approach and urge linguistically motivated ontology development, whereby ontologies would contain human-readable linguistic resources that can offer strong evidence in the mapping process. To facilitate this process, the OntoLing plug-in [20] was developed for the Protégé editor. However, as pointed out by the authors, this enrichment process is currently unstandardised. As a result, it can be difficult to build CLOM systems based upon such linguistically enriched ontologies. Jung et al. [19] demonstrate indirect alignment for multilingual ontologies in English, Korean and Swedish, given alignment A which is generated between ontology O_1 (i.e. in Korean) and O_2 (i.e. in English), and alignment A' which is generated between ontology O_2 and O_3 (i.e. in Swedish), mappings between O_1 and O_3 can be generated by reusing alignment A and A' since they both concern one common ontology O_2. Assuming the availability of A and A', this is an achievable approach. However, as this technique requires the very existence of A and A' which currently remains a challenge in itself, it can be difficult to apply this approach in some CLOM settings.

Translating ontology labels is a popular technique to convert a cross-lingual mapping problem into a monolingual mapping problem. Bilingual dictionaries, multi-lingual thesauri and off-the-shelf machine translation (MT) tools are often used as media to bridge between different natural languages presented in the ontologies at hand. Zhang et al. [12] use a Japanese-English dictionary to translate the labels in the Japanese web directory into English first, before carrying out monolingual matching procedures using the RiMOM tool in the OAEI 2008 mldirectory[1] test case. Bouma [21] uses the multilingual EuroWordNet and the Dutch Wikipedia to align the GTAA thesaurus (in Dutch) to Wordnet and DBpedia (both in English). Wang et al. [10] use the GoogleTranslate service to translate digital library vocabularies before applying instance-based matching techniques to generate mappings among library subjects written in English, French and German. Trojahn et al. [13] incorporate the work presented in [14, 19] and uses the GoogleTranslate API as the translation medium to achieve CLOM. In addition, their tool is accompanied by a mapping reuse feature as proposed in [19]. The aforementioned research illustrates that translation can serve as a means to the completion of CLOM tasks, and MT may be sufficient to bridge between different natural languages in a given CLOM scenario, but just how suitable are these translations in the matching sense as opposed to the linguistic sense? This question is discussed in detail next.

3 The Challenge of Translation in CLOM

In the well studied field of MT, various techniques aiming to improve the quality of translation such as statistical MT, rule-based MT are designed, all equipped with the ability to disambiguate word senses. By nature, MT tools are intended to generate the most accurate translations in the linguistic sense, which is not necessarily a requirement in CLOM. This is because ontology matching techniques often rely on the discovery of lexical similarities as demonstrated in [14]. To achieve CLOM, translation is merely a stepping-stone to the actual goal which is generating correspondences between ontological entities. Consequently, translating source ontology labels is not centred around finding localised equivalents for them, but to select translations that can lead them to quality candidate matches in the target ontology. A translation may be accurate in the eyes of a linguist (i.e. linguistically correct), but it may not be appropriate (i.e. neglect matches) in the mapping context. In this paper, *an appropriate ontology label translation (AOLT) in the context of cross-lingual ontology mapping is one that is most likely to maximize the success of the subsequent monolingual ontology matching step*. This notion of AOLT in CLOM can be illustrated in the following example where the source ontology is in English and the target ontology is in French. A source concept *Ph.D. Student* has a candidate translation *Ph.D. Étudiant* which has a synonym *Étudiant au doctorat* (for example, by looking up from a thesaurus). The target ontology happens to have a class labelled *Étudiant au doctorat*, in this case, *Étudiant au doctorat* should be considered as the AOLT in this scenario since it is the terminology used by the target ontology and is most likely to lead to a mapping as a result.

Note that the work presented in this paper should not be confused with ontology localisation, whereby ontology labels are translated so that the given ontology is adapted "to a particular language and culture" [22]. In this paper, ontology labels are

[1] `http://oaei.ontologymatching.org/2008/mldirectory`

purposely translated so that the given ontologies can be best mapped. The AOLT process is concerned with searching for appropriate translations (from a mapping point of view) among a pool of candidate translations that are believed to be the ones most likely to enhance the matching ability of the subsequent monolingual matching step, but not necessarily the most linguistically correct translations (from a localisation point of view). Note this is not a natural language processing technique, the AOLT process does not attempt to disambiguate word senses.

4 Using Pseudo Feedback in CLOM

Ruthven & Lalmas [23] present an extensive survey on relevance feedback used in information retrieval (IR). Broadly speaking, there are three types of relevance feedback, explicit, implicit and blind (also known as pseudo feedback). Explicit feedback is obtained after a query is issued by the user and an initial set of documents is retrieved, the user marks these initial documents as relevant or not relevant, and the system retrieves a better list of documents based on this feedback by computing a single or multiple iterations. Implicit feedback works similarly but attempts to infer users' intentions based on observable behaviour. Pseudo feedback is generated when the system makes assumptions on the relevancy of the retrieved documents. Explicit user feedback in monolingual ontology matching and its effectiveness is successfully demonstrated by Duan et al. in [24], where the user marks the matches generated to be true or false. This paper expands on the feedback techniques that can be used in ontology mapping which is inspired by pseudo feedback in IR, it concerns a feedback mechanism without the involvement of a user in cross-lingual mapping scenarios.

When using feedback in the context of CLOM, an initial set of matches generated after the first iteration of the CLOM process can be thought of as the initial set of documents retrieved by an IR system, and the assumption made against document relevancy in IR becomes the process of assuming which candidate matches in the initial set are indeed correct. Similarity measures are often used to illustrate the confidence level of a matching tool in its conclusion of a matched entity pair, which can be used by pseudo feedback when making assumptions on correct matches. There are many types of similarity measures used in ontology matching as documented by Euzenat & Shvaiko [25]. Although currently there is no obvious method that is a clear success [26], similarity measures nonetheless are a way to perceive the probability of a match being correct or not. The CLOM approach presented in this paper incorporates pseudo feedback, whereby the system assumes after an initial execution that matches with confidence measures above a certain threshold are correct. It then examines how these matches are generated. Currently, this involves examining which translation media were used. The results of this examination then influence the selection of AOLTs in the second iteration of the system. An overview of this approach is presented in section 4.1, followed by its implementation in section 4.2.

4.1 Process Overview

Fig. 1 illustrates the CLOM process that integrates pseudo feedback. Given ontologies O_1 and O_2 that are labelled in different natural languages, CLOM is achieved in three main steps. Firstly, O_1 is transformed through the *ontology rendering* process as O_1', which has the same structure as O_1 but contains entities labelled in the natural language that is used by O_2. Secondly, O_1' is matched to O_2 using *monolingual ontology matching* techniques to generate candidate matches. Thirdly, these matches are reviewed by the *match assessment* process, where assumptions are made to speculate on correctness. The pseudo feedback, containing the translation media used by these "correct" matches, is then processed by the AOLT selection in the second iteration of the CLOM system. Each of these steps is discussed next in detail.

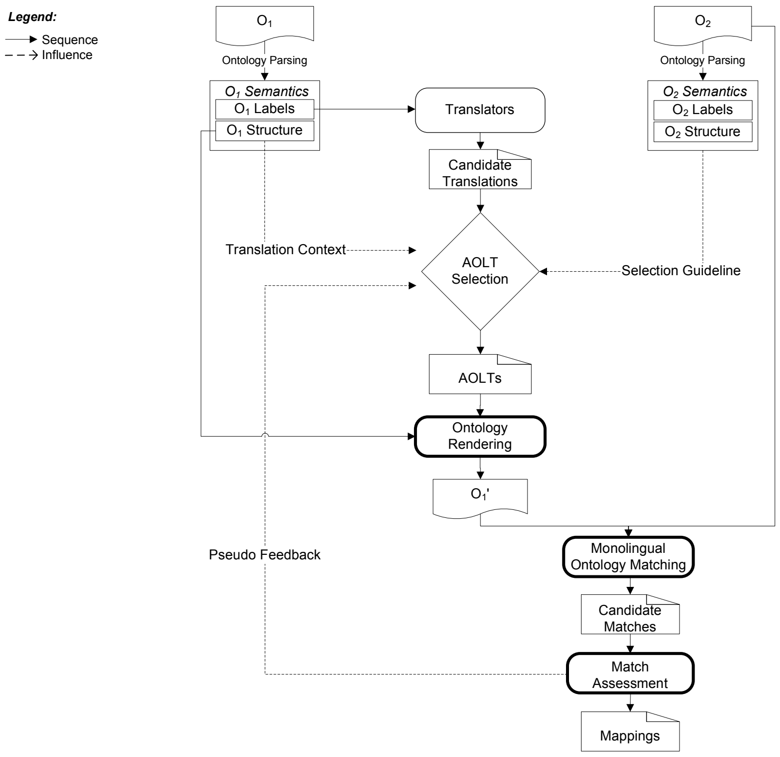


Fig. 1. Pseudo Feedback in CLOM

Ontology renditions are achieved by structuring the translated labels[2] in the same way as the original ontology, and assigning them with new base URIs to create well-formed ontology resources[3]. Zhao et al. [27] define ontology rendition as a process in

[2] In this paper, the translation of ontology labels refers to the translation of strings that are used to identify ontological resources in a formally defined ontology, e.g. the value of `rdf:ID` in `<Class rdf:ID="Thing"/>` or the fragment identifier, i.e. the string after the hash sign in `<owl:Class rdf:about="http://swrc.ontoware.org/ontology#Person"/>`. It does not refer to the content of `rdfs:label` elements such as `<rdfs:label>Thing</rdfs:label>`.

[3] The base URI is the unique identifier for an ontology and the resources within, as the resources in O_1' should not point to the original resources in O_1, new namespace declarations are assigned to the translated labels.

the ontology development that consists of two roles, converting and interpreting. The converting role is the transformation of an ontology where the output has "formally different but theoretically equivalent" semantics, e.g. translating ontologies from OWL to RDF via Web-PDDL [28]. The interpreting role renders formally specified commitments, which is the aim of the ontology rendering process shown in Fig. 1. Note that the structure of an input ontology is not changed during this process, as doing so would effectively alter the semantics of the original ontology.

The AOLT selection process makes use of the pre-defined semantics in the given ontologies and is concerned with identifying the most appropriate translations for a specific mapping scenario. To achieve this, firstly, for each extracted label in O_1, it is sent to the *translators* to generate candidate translations. Secondly, to identify the AOLTs in the specified CLOM scenario for an ontology label, the selection process takes the following into account:

- The *semantics in O_1* indicate the context that a to-be-translated label is used in. Given a certain position of the node with this label, the labels of its surrounding nodes can be collected to represent the context of use. For example, for a class node, its context can be represented by the labels of its super/sub/sibling-classes. For a property node, its context can be represented by the labels of the resources which this property restricts. For an individual of a class, its context can be characterised by the label of the class it belongs to.
- As O_1' is rendered so that its representation of O_1 can be best mapped to O_2, the *semantics in O_2* therefore act as broad AOLT selection guidelines. For example, when several translation candidates are available for a label in O_1, the most appropriate translation is the one that is most similar to what is used in O_2, e.g. the example given in section 3.

Once AOLTs are identified, O_1'[4] is generated and various monolingual matching techniques can be applied to create correspondences between O_1' and O_2. These matches are finally sent to the *match assessment* process, and "correct" matches are assumed to be those that have confidence measures above a specified threshold. Based on this assumption, pseudo feedback is generated which contains the most effective translation media for the particular ontologies at hand. In the second iteration of the CLOM system, the translations returned from these media are perceived to be the AOLTs. This process is further demonstrated and explained with an example in section 5.1.

4.2 Implementation

An implementation of the proposed approach is shown in Fig. 2. The Jena Framework[5] 2.5.5 is used to parse the ontologies, extract entity labels and to generate surrounding resource labels for a given entity. Candidate translations of the source ontology labels are obtained from the *machine translation service* that uses the GoogleTranslate[6] API 0.5 and the WindowsLive[7] translator. These candidate translations are stored in the

[4] O_1' exists only for purpose of the mapping, it should not be considered as a localised O_1.
[5] `http://jena.sourceforge.net`
[6] `http://code.google.com/p/google-api-translate-java`
[7] `http://www.windowslivetranslator.com/Default.aspx` Note at the time of writing, the Windows Live translator has been renamed as the Bing translator.

translation repository and formatted in XML. For the ontology pair shown in Fig. 3, Fig 3a presents a snippet of the translation repository generated for the source ontology labelled in English, and Fig. 3b shows a snippet of the lexicon repository generated for the target ontology labelled in French. Ontology labels are often concatenated (as white spaces are not allowed in the OWL/RDF naming conversion), which cannot be processed by the integrated MT tools. To overcome this issue, concatenated ontology labels (stored in the *OntLabel* attribute in Fig. 3) are first split into sequences of their constituent words (as machine readable values and stored in the *MRLabel* attribute in Fig. 3) before passed to the MT tools. In the example shown in Fig. 3, as capital letters are used to indicate the beginning of another word, white spaces are inserted before each capital letter found other than the first one. A *lexicon repository* is generated that contains the target ontology labels, their corresponding synonyms and surroundings. An example of this is shown in Fig. 3b. Synonyms are generated by calling the *lexicon dictionary service*, which queries synonyms-fr.com[8] for synonyms in French. The generations of the translation repository and the lexicon repository take place in parallel. Finally, both repositories are stored in the eXist DB[9] 1.0rc.

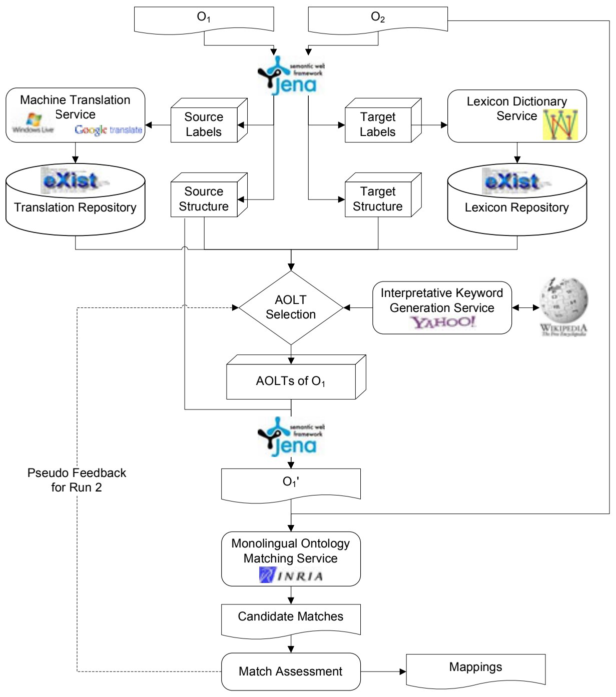
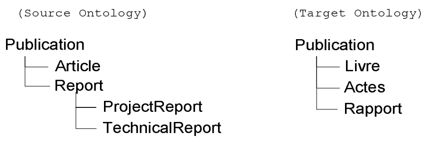


Fig. 2. An Implementation of Pseudo Feedback in CLOM

The AOLT selection process queries the repositories to compare the candidate translations of an ontology label to the resources stored in the lexicon repository. If

8 `http://synonyms-fr.com`
9 `http://exist.sourceforge.net`

matches (to a target label or synonym of a target label) are found, preference is always given to what is used by the target ontology (e.g. when a candidate translation is linked to a target label's synonym, this synonym's corresponding label that appears in the target ontology is deemed to be the AOLT). If no match is found, for each candidate, a set of interpretative keywords are generated to illustrate the meaning of this candidate. This is achieved by querying Wikipedia[10] via the Yahoo Term Extraction Tool[11]. Using a space/case-insensitive edit distance string comparison algorithm based on Nerbonne et al.'s method [29], the candidate with keywords that are most similar to the source label's semantic surrounding is chosen as the AOLT. Finally, AOLTs are concatenated to construct well-formed resource labels (stored as values in the attribute *ConLabel* of the *Candidate* element in Fig. 3a and the *Syn* element in Fig. 3b) by replacing white spaces with underscores. Once the AOLTs are determined for source labels, given the original ontology structure, O_1' is generated using the Jena framework, and matched to O_2 using the Alignment API[12] 3.6.



```
…
<Resource id="CLS-1" OntLabel="Article" MRLabel="Article"/>
    <Translation>
        <Candidate id="CDD-0" value="L'article" source="google" ConLabel="L'article"/>
        <Candidate id="CDD-1" value="Article" source="wl" ConLabel="Article"/>
    </Translation>
    <Surrounding id="CLS-0" OntLabel="Publication"/>
    <Surrounding id="CLS-2" OntLabel="Report"/>
</Resource>
…
<Resource id="CLS-3" OntLabel="ProjectReport" MRLabel="Project Report"/>
    <Translation>
        <Candidate id="CDD-7" value="Rapport de projet" source="google"
        ConLabel="Rapport_de_projet"/>
        <Candidate id="CDD-8" value="Rapport de projet" source="wl"
        ConLabel="Rapport_de_projet"/>
    </Translation>
    <Surrounding id="CLS-2" OntLabel="Report"/>
    <Surrounding id="CLS-4" OntLabel="TechnicalReport"/>
</Resource>
…
```

(a) Translation Repository – An Example

```
…
<Resource id="CLS-0" OntLabel="Publication" MRLabel="Publication"/>
    <Synonym>
        <Syn id="SYN-0" value="Parution" source="synonyms-fr.com" ConLabel="Parution"/>
        <Syn id="SYN-1" value="Sortie" source="synonyms-fr.com" ConLabel="Sortie"/>
        <Syn id="SYN-2" value="Ouvrage" source="synonyms-fr.com" ConLabel="Ouvrage"/>
    </Synonym>
    <Surrounding id="CLS-1" OntLabel="Livre"/>
    <Surrounding id="CLS-2" OntLabel="Actes"/>
    <Surrounding id="CLS-3" OntLabel="Rapport"/>
</Resource>
…
```

(b) Lexicon Repository – An Example

Fig. 3. Examples of the Translation Repository & the Lexicon Repository

[10] http://www.wikipedia.org
[11] http://developer.yahoo.com/search/content/V1/termExtraction.html
[12] http://alignapi.gforge.inria.fr

Collisions can occur when more than one entity in O_1 concludes with the same value as its AOLT. A summary of collision solutions is presented in Table 1. When a collision is detected between two entities, priority is given to the one that was influenced by the target ontology (e.g. derived based on a match to target label or synonym) as scenario *i*, *ii*, *iii* and *iv* illustrate in Table 1. If both entities arrive to the same AOLT with an equal strategy (e.g. both came from a match made to a target label's synonym) as shown in Table 1 scenario *v*, *vi* and *vii*, the later entity will seek the next translation in line. If no more alternative translation is available to this later entity, a numerical number (that is checked to be free of collision) is attached to the collided term. This ensures that both entities will have well-formed (i.e. unique) URIs. These numbers are selected at random with the intent of avoiding the introduction of any kind of patterns into the translation selection process.

Table 1. Resolving Translation Collision

Collision Scenario	AOLT Selection Strategy		Solution
	Entity 1	Entity 2	
i	candidate translation matches target label's synonym	candidate translation matches target label	entity 2 keeps the collided AOLT; entity 1 seeks alternative translation
ii	derived from interpretative keyword comparison	candidate translation matches target label's synonym	
iii	candidate translation matches target label	candidate translation matches target label's synonym	entity 1 keeps the collided AOLT; entity 2 seeks alternative translation
iv	candidate translation matches target label's synonym	derived from interpretative keyword comparison	
v	candidate translation matches target label	candidate translation matches target label	
vi	candidate translation matches target label's synonym	candidate translation matches target label's synonym	
vii	derived from interpretative keyword comparison	derived from interpretative keyword comparison	

Given the matching results[13] generated by the Alignment API and the origins of all AOLTs, the *match assessment* process assumes that matches with at least 0.5 confidence levels are correct[14] and computes a set of statistical feedback based on this assumption. This feedback contains the usage (as percentages) of each translation medium used by the "correct" matches. The translations which are generated by the highest ranked (i.e. highest usage) MT tools are prioritised in the AOLT selection process during the second execution of the system. This is further illustrated with an example in section 5.1.

[13] In the Alignment API, a match between a source ontology resource and a target ontology resource is represented with a relation and accompanied by a confidence level that range between 0 (not confident) and 1 (confident).

[14] As confidence levels range between 0 and 1, 0.5 is a natural division point where matches would either incline towards being either confident (i.e. equal or above 0.5) or not confident (i.e. below 0.5). This threshold on confidence measure cannot be configured by the user in the current implementation, as this paper is a proof of concept of whether pseudo feedback can be applied in CLOM rather than looking for a best feedback configuration in CLOM. The pseudo feedback presented in this paper speculates on which matches could be correct, it is not designed as an accurate assessment of the matches generated.

5 Evaluation

To evaluate the effectiveness of the proposed pseudo feedback mechanism, the implemented system is compared to a baseline system that is already proven effective in [15]. The only difference between the two approaches is the pseudo feedback. The baseline system integrates the same set of tools and APIs, except that it does not attempt to use pseudo feedback to influence the selection of AOLTs (i.e. the baseline system is the first iteration of the implementation discussed in section 4.2). The evaluation experiment uses the OAEI 2009 benchmark dataset involving ontologies of the bibliography domain labelled in English and French[15]. Its setup is discussed in section 5.1, followed by its findings in section 5.2.

5.1 Experimental Setup

Fig. 4 illustrates an overview of the experimental setup. Ontology 101 is labelled in English and has 36 classes, 24 object properties, 46 data type properties and 137 instances. Ontology 206 contains similar semantics, except it has one less object property and is labelled in French. The English ontology is matched to the French ontology using the proposed approach with pseudo feedback and the baseline approach to generate mappings M_F and M_B respectively, using eight matching algorithms[16] that are supported by the Alignment API.

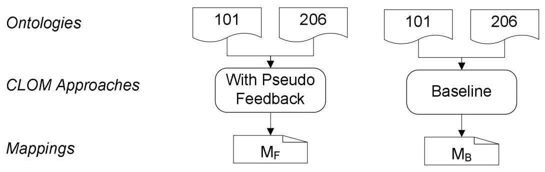


Fig. 4. Experiment Overview

Note that the original OAEI test scenario does not involve any translations of ontology labels. It was designed to examine the strength of structure-based monolingual matching techniques since ontology 101 and 206 have highly similar structures. Though this is not the goal of the evaluation setup presented in this paper, nevertheless, this test case provides us with a pair of ontologies in different natural languages and a reliable gold standard[17] for the evaluation of M_F and M_B.

For each matching algorithm executed, the pseudo feedback mechanism selects the matches with at least 0.5 confidence levels, and investigates how the AOLTs were determined among these "correct" matches. For example, the pseudo feedback generated after the first iteration of the system when using the StrucSubDist-Alignment matching algorithm is shown in Fig. 5. The attributes of the root element

[15] `http://oaei.ontologymatching.org/2009/benchmarks`

[16] The algorithms used in the experiment are NameAndPropertyAlignment, StrucSubsDist-Alignment, ClassStructAlignment, NameEqAlignment, SMOANameAlignment, SubsDist-NameAlignment, EditDistNameAlignment and StringDistAlignment.

[17] `http://oaei.ontologymatching.org/2009/benchmarks/206/refalign.rdf`

include the matching algorithm used (stored in the `algorithm` attribute in Fig. 5), the cut-off point of the assumption (stored in the `threshold` attribute in Fig. 5), the total matches generated by the specified matching algorithm (stored in the `matches` attribute in Fig. 5) and the assumed-to-be correct matches found (stored in the `estimate` attribute in Fig. 5). In the case for the StrucSubsDistAlignment matching algorithm shown in Fig. 5, at a threshold of 0.5, a total of 86 correct matches are identified within a set of 103 matches. Each `<Entry>` element records the total count (stored in the `count` attribute in Fig. 5) of a particular translation medium used (stored in the `medium` attribute in Fig. 5) and its accumulated usage (stored in the `usage` attribute in Fig. 5, which is calculated as *count/estimate*). In Fig. 5, the pseudo feedback indicates that firstly, the majority of AOLTs originated from the target ontology (i.e. either labels used in the target ontology or synonyms of these labels). Secondly, it shows that the same translations were returned by the integrated MT tools at times. In such cases, it would not be fair to credit either MT tool, it is therefore categorised on its own (ranked second highest in the example shown in Fig. 5). Thirdly, a greater number of AOLTs came from the GoogleTranslate API (in third rank) than the WindowsLive translator (in fourth rank) when using the StrucSubsDist-Alignment algorithm in this particular experiment. Finally, it shows that a small number of matches are made between externally defined resources (e.g. `rdf:resource ='http://www.w3.org/1999/02/22-rdf-syntax-ns#List'` as defined by the World Wide Web Consortium) which are categorised in fifth.

```
<PseudoFeedback algorithm="StrucSubsDistAlignment" threshold="0.5" matches= "103.0"
estimate="86.0" >
    <Entry count="31.0" medium="TargetOntology" usage="0.360465"/>
    <Entry count="23.0" medium="BothMT" usage="0.267441"/>
    <Entry count="17.0" medium="Google" usage="0.197674"/>
    <Entry count="12.0" medium="WindowsLive" usage="0.139534"/>
    <Entry count="3.0" medium="External" usage="0.034883"/>
</PseudoFeedback>
```

Fig. 5. An Example of Pseudo Feedback

In the second iteration of the system using the StrucSubsDistAlignment algorithm, the strategy for which translation media to use is thus determined by the order shown in Fig. 5. For a source ontology label, when a translation that originates from the target ontology is available, it is chosen as the AOLT; if not, use the translation that is agreed by both MT tools; in the absence of these two options, choose the translation returned from the GoogleTranslate API; if all fails, use the translation returned from the WindowsLive translator. This feedback to the AOLT selection process is repeated for all other matching algorithms in the second run of the system. Note that the ranking of the translation media is not necessarily always the same with what is shown in Fig. 5, as it depends on the statistics generated by the pseudo feedback which varies by the matching techniques used.

In addition to what is discussed in Table 1, new rules are included in the collision resolution process for the second iteration of the system. Priority is given to higher ranked MT media. For example, when two entities both choose the same value as its AOLT, the system checks how they each arrived to this conclusion. The higher ranked translation strategy will keep the collided term as its AOLT, and the other entity will seek for an alternative from a lower ranked MT medium. It is possible that a collision is unsolved still when all other alternatives cause further collisions or simply do not exist. In such cases, for the entity that is seeking an alternative translation, a unique numerical number is attached to the end of collided term as explained previously in section 4.2.

5.2 Experimental Results

M_F and M_B were evaluated based on the gold standard (see footnote 15) provided by the OAEI. Firstly, the evaluation identifies the correct matches in M_F and M_B based on the gold standard, computes and compares their respective precision, recall and F-measure scores. A correct mapping is one that is included in the gold standard regardless of its confidence measure. Precision (shown in Fig. 6a), recall (shown in Fig. 6b) and F-measure (shown in Fig. 6c) scores were calculated for all eight experimented matching algorithms. Fig. 6 shows higher precision, recall and F-measure scores for the matches found in M_F across all matching algorithms. On average, M_B has a precision of 0.7355, a recall of 0.5928 and an F-measure of 0.6428, which have all been improved when the pseudo feedback mechanism is incorporated, leading to an average precision of 0.7875, recall of 0.6268 and F-measure of 0.6873 in M_F. These statistics indicate that M_F not only contains a greater number of correct matches, but also is more complete than M_B.

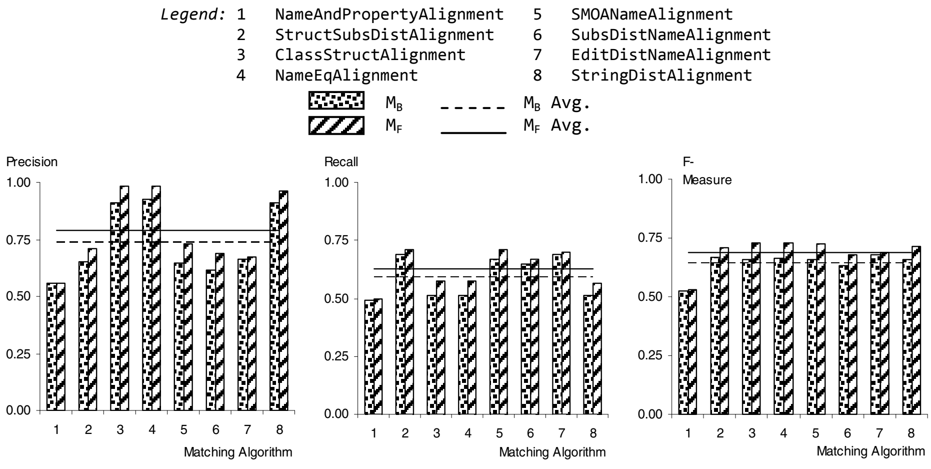


Fig. 6. Precision, Recall and F-Measure Overview

Secondly, as confidence levels are not accounted by precision, recall or F-measure, for the correct matches found in M_B and M_F, their confidence means and standard deviations were also calculated. The mean is the average confidence of the correct matches found in a set of matches, where higher means indicate more confident results. The standard deviation is a measure of dispersion, where the greater it is, the bigger the spread in the confidence levels. Higher quality matches therefore are those with high confidence means and low standard deviations. Note, some matching

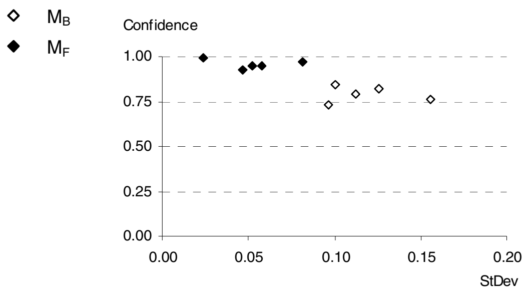


Fig. 7. Confidence Measures in M_B and M_F

algorithms (e.g. NameEqAlignment, StringDistAlignment and ClassStruct-Alignment which incorporated StringDistAlignment in the experiment) only created matches with 1.0 confidence levels, therefore were not included in this study. Fig. 7 presents an overview on the evaluation of the confidence means and standard deviations. The correct matches in M_F are always higher in confidence and lower in dispersion. This finding indicates that in addition to improving the precision, recall and F-measure, the pseudo feedback feature can also facilitate monolingual matching techniques in their ability to generate correct matches more confidently.

It may be argued that as the differences shown in the F-measure scores between M_B and M_F are relatively small, it would be difficult to conclude an improvement in M_F. To validate the statistical significance of the findings so far, and to validate the difference (if it exists) between the two approaches, paired t-tests were carried out on the F-measure scores across all matching algorithms and a p-value of 0.003 is found. At a significance level of α=0.05, this p-value rejects the null hypothesis (null hypothesis being there is no difference between the two CLOM approaches) and indicates that the findings are statistically significant. This further confirms the effectiveness of the pseudo feedback mechanism in the experiment.

It should be noted that the experimental setup is somewhat limited in the size of the ontologies used, their comparable natural languages and structures (as discussed in section 5.1). Nonetheless, the evaluation results from this experiment do suggest a positive impact of pseudo feedback and its ability to facilitate monolingual ontology matching techniques in the process of generating quality CLOM results.

6 Conclusions and Future Work

This paper presents a novel approach to CLOM that incorporates the pseudo feedback technique that is similar to the well-established relevance feedback mechanism used in the field of information retrieval. The proposed approach makes assumption on the correct matches in an initial matching set that is generated after the first iteration of the CLOM system. The pseudo feedback mechanism then determines how these "correct" matches are generated by detecting the translation media used, and finally sends this feedback back to the system to aid the selection of ontology label translations in the second iteration of the CLOM system. The advantages of the proposed approach are demonstrated using an OAEI dataset and evaluated against the

OAEI gold standard. Based on the experimental findings presented in this paper, there are indications that the proposed pseudo feedback feature enhances the performance of the monolingual ontology matching techniques used.

Several potential future research directions can be derived from the findings presented in this paper. Firstly, the use of feedback in CLOM can be expanded to incorporate explicit and implicit feedback, whereby user knowledge and user behaviours may be used to assist the generation of reliable mappings. Secondly, current implementation of the proposed approach in this paper can be extended. For example, the current pseudo feedback mechanism assumes that correct matches are above the 0.5 confidence level, future implementations may include several thresholds that can be configured by users. The pseudo feedback can also be further extended to implement negative feedback (i.e. a blacklist as opposed to a whitelist of translation media as shown in this paper) so that the AOLT selection process recognises what not to do in a given mapping scenario. Additionally, MT tools in the current implementation are not specialised to work with highly refined domains such as medical ontologies. This may be improved given domain-specific translation tools. Thirdly, the risks involved and their impact (e.g. when the assumptions made on the "correct" matches are simply invalid) on the CLOM quality when applying pseudo feedback in CLOM is not yet investigated in this paper, future research could explore this area. Fourthly, only two iterations of the CLOM system is demonstrated in this paper, further iterations of the system using pseudo feedback can be evaluated in order to investigate whether a third, fourth etc. iteration of the process can further improve mapping quality. Lastly, the ontologies used in the experiment shown in this paper are relatively small in size of the same domain, with comparatively similar natural language pairs and structures, the scalability and the effectiveness of the proposed approach should be tested against large ontology pairs with overlapping domains that involve more distinct natural languages and structures. This is currently being investigated as part of the on-going research.

Acknowledgement. This research is partially supported by Science Foundation Ireland (Grant 07/CE/I1142) as part of the Centre for Next Generation Localisation (www.cngl.ie), and the Higher Education Authority (PRTLI4 fund) as part of Networked Embedded Systems project (www.nembes.org) at Trinity College Dublin.

References

1. Nirenburg, S., Raskin, V.: Ontological Semantics, Formal Ontology, and Ambiguity. In: Proceedings of the 2nd International Conference on Formal Ontology in Information Systems, pp. 151–161 (2001)
2. Bateman, J.A.: Ontology Construction and Natural Language. In: Workshop on Formal Ontology in Conceptual Analysis and Knowledge Representation (1993)
3. Noy, N.F., McGuinness, D.L.: Ontology development 101: A guide to creating your first ontology. Technical Report SMI-2001-0880, Stanford Medical Informatics (2001)
4. Caracciolo, C., Sini, M., Keizer, J.: Requirements for the Treatment of Multilinguality in Ontologies within FAO. In: Proceedings of the 3rd International Workshop on OWL: Experiences and Directions (2007)

5. Kerremans, K., Temmerman, R., Tummers, J.: Representing Multilingual and Culture-Specific Knowledge in a VAT Regulatory Ontology: Support from the Termontography Method. In: Chung, S. (ed.) OTM-WS 2003. LNCS, vol. 2889, pp. 662–674. Springer, Heidelberg (2003)
6. Chen, H., Wang, Y., Wang, H., Mao, Y., Tang, J., Zhou, C., Yin, A., Wu, Z.: Towards a Semantic Web of Relational Databases: a Practical Semantic Toolkit and an In-Use Case from Traditional Chinese Medicine. 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. 750–763. Springer, Heidelberg (2006)
7. Nichols, E., Bond, F., Tanaka, T., Fujita, S., Flickinger, D.: Multilingual Ontology Acquisition from Multiple MRDs. In: Proceedings of the 2nd Workshop on Ontology Learning and Population, pp. 10–17 (2006)
8. Buitelaar, P., Cimiano, P., Haase, P., Sintek, M.: Towards Linguistically Grounded Ontologies. 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. 111–125. Springer, Heidelberg (2009)
9. Peters, W., Montiel-Ponsoda, E., Aguado de Cea, G.: Localizing Ontologies in OWL. In: Proceedings of the OntoLex 2007 Workshop (2007)
10. Wang, S., Isaac, A., Schopman, B., Schlobach, S., Van der Meij, L.: Matching Multilingual Subject Vocabularies. In: Agosti, M., Borbinha, J., Kapidakis, S., Papatheodorou, C., Tsakonas, G. (eds.) ECDL 2009. LNCS, vol. 5714, pp. 125–137. Springer, Heidelberg (2009)
11. Trojahn, C., Quaresma, P., Vieira, R.: A Framework for Multilingual Ontology Mapping. In: Proceedings of the 6th edition of the Language Resources and Evaluation Conference, pp. 1034–1037 (2008)
12. Zhang, X., Zhong, Q., Li, J., Tang, J., Xie, G., Li, H.: RiMOM Results for OAEI 2008. In: Proceedings of the 3rd International Workshop on Ontology Matching, pp. 182–189 (2008)
13. Trojahn, C., Quaresma, P., Vieira, R.: An API for Multi-lingual Ontology Matching. In: Proceedings of the 7th Conference on International Language Resources and Evaluation, pp. 3830–3835 (2010) ISBN 2-9517408-6-7
14. Fu, B., Brennan, R., O'Sullivan, D.: Cross-Lingual Ontology Mapping – An Investigation of the Impact of Machine Translation. In: Gómez-Pérez, A., Yu, Y., Ding, Y. (eds.) ASWC 2009. LNCS, vol. 5926, pp. 1–15. Springer, Heidelberg (2009)
15. Fu, B., Brennan, R., O'Sullivan, D.: Cross-Lingual Ontology Mapping and Its Use on the Multilingual Semantic Web. In: Proceedings of the 1st Workshop on the Multilingual Semantic Web, CEUR, vol. 571, pp. 13–20 (2010)
16. Liang, A., Sini, M.: Mapping AGROVOC & the Chinese Agricultural Thesaurus: Definitions, Tools Procedures. New Review of Hypermedia & Multimedia 12(1), 51–62 (2006)
17. Ngai, G., Carpuat, M., Fung, P.: Identifying Concepts Across Languages: A First Step towards A Corpus-based Approach to Automatic Ontology Alignment. In: Proceedings of the 19th International Conference on Computational Linguistics, vol. 1, pp. 1–7 (2002)
18. Pazienta, M., Stellato, A.: Linguistically Motivated Ontology Mapping for the Semantic Web. In: Proceedings of the 2nd Italian Semantic Web Workshop, pp. 14–16 (2005)
19. Jung, J.J., Håkansson, A., Hartung, R.: Indirect Alignment between Multilingual Ontologies: A Case Study of Korean and Swedish Ontologies. In: Håkansson, A., Nguyen, N.T., Hartung, R.L., Howlett, R.J., Jain, L.C. (eds.) KES-AMSTA 2009. LNCS (LNAI), vol. 5559, pp. 233–241. Springer, Heidelberg (2009)

20. Pazienza, M.T., Stellato, A.: Exploiting Linguistic Resources for Building Linguistically Motivated Ontologies in the Semantic Web. In: Proceedings of OntoLex Workshop 2006: Interfacing Ontologies and Lexical Resources for Semantic Web Technologies (2006)
21. Bouma, G.: Cross-lingual Ontology Alignment using EuroWordNet and Wikipedia. In: Proceedings of the 7th Conference on International Language Resources and Evaluation (LREC 2010), pp. 1023–1028 (2010) ISBN 2-9517408-6-7
22. Suárez-Figueroa, M.C., Gómez-Pérez, A.: First Attempt Towards A Standard Glossary of Ontology Engineering Terminology. In: Proceedings of the 8th International Conference on Terminology and Knowledge Engineering (2008)
23. Ruthven, I., Lalmas, M.: A survey on the use of relevance feedback for information access systems. Knowledge Engineering Review 18(2), 95–145 (2003)
24. 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 II. LNCS, vol. 6497. Springer, Heidelberg (2010)
25. Euzenat, J., Shvaiko, P.: Ontology Matching. Springer, Berlin (2007)
26. Ichise, R.: Evaluation of Similarity Measures for Ontology Mapping. In: Hattori, H., Kawamura, T., Idé, T., Yokoo, M., Murakami, Y. (eds.) JSAI 2008. LNCS (LNAI), vol. 5447, pp. 15–25. Springer, Heidelberg (2009)
27. Zhao, G., Zheng, J., Meersman, R.: An Architecture Framework for Ontology Development. In: Proceedings of the IADIS International Conference (2003)
28. Dou, D., McDermott, D., Qi, P.: Ontology Translation on the Semantic Web. Journal on Data Semantics II, 35–57 (2004)
29. Nerbonne, J., Heeringa, W., Kleiweg, P.: Edit Distance and Dialect Proximity. In: Time Warps, String Edits and Macromolecules: The Theory and Practice of Sequence Comparison, 2nd edn., pp. v–xv. CSLI, Stanford (1999)

Automatic Identification of Ontology Versions Using Machine Learning Techniques

Carlo Allocca

Knowledge Media Institute (KMi), The Open University, Walton Hall,
Milton Keynes MK7 6AA, United Kingdom
c.allocca@open.ac.uk

Abstract. When different versions of an ontology are published online, the links between them are often lost as the standard mechanisms (such as owl:versionInfo and owl:priorVersion) to expose these links are rarely used. This generates issues in scenarios where people or applications are required to make use of large scale, heterogenous ontology collections, implicitly containing multiple versions of ontologies. In this paper, we propose a method to detect automatically versioning links between ontologies which are available online through a Semantic Web search engine. Our approach is based on two main steps. The first step selects candidate pairs of ontologies by using versioning information expressed in their identifiers. In the second step, these candidate pairs are characterized through a set of features, including similarity measures, and classified by using Machine Learning Techniques, to distinguish the pairs that represent versions from the ones that do not. We discuss the features used, the methodology employed to train the classifiers and the precision obtained when applying this approach on the collection of ontologies of the Watson Semantic Web search engine.

1 Introduction

Ontologies evolve according to the modifications carried out in response to changes in the represented domain or its conceptualization [20]. Such an evolution of an ontology, O_i, produces a finite collection of documents $\{O_{i1}, O_{i2},...,O_{im}\}$, where each O_{ij} represents a particular version of O_i. Although, mechanisms exist to keep track of the links between different versions (e.g. owl:versionInfo, owl:priorVersion), ontologies that are exposed on the Web rarely make use of such ways to link versions [3] (e.g. `http://lsdis.cs.uga.edu/projects/semdis/sweto/testbed_v1_3.owl` and `http://lsdis.cs.uga.edu/projects/semdis/sweto/testbed_v1_4.owl`).

Semantic Web search engine (SWSEs) systems, such as Watson[1], Swoogle[2] or Sindice[3] collect and index ontologies in order to facilitate their reuse [1]. However, the lack of explicit links between versions of ontologies hampers the various scenarios where users or applications make use of such large scale, heterogenous

[1] http://watson.kmi.open.ac.uk
[2] http://swoogle.umbc.edu/
[3] http://sindice.com/

© Springer-Verlag Berlin Heidelberg 2011

collection of ontologies, implicitly containing multiple ontology versions [7]. In particular, users might require the most up-to-date information or need to compare different versions of an ontology, to be able to choose the most appropriate one.

In this paper, we investigate an approach *to automatically detect/discover implicit versioning links in large ontology collections, in order to make them explicit*. In contrast with the approaches described in [15,14,19,12,21,23] (see Section 5), we propose a version detection mechanism for ontologies where versioning metadata are not available and the ontologies are collected and indexed by a SWSE system. To this end, two important issues need to be addressed: *(1) how to select pairs of ontologies as candidate versions*; *(2) how to decide whether the candidate pairs are versions or not*. To tackle (1), we adopt a solution based on identifying and extracting versioning information codified in the ontology URIs. To deal with (2), we propose a set of features which characterize ontology versions and we apply machine learning techniques to classify the candidate ontology pairs as *PrevVersion* and *NotPrevVersion*. The method has been evaluated using the Watson collection of OWL ontologies, which is a dataset of 7000 ontologies. The evaluation shows that the features we considered provide an accurate characterization of ontology versions, leading to high precision when used in three different types of classifiers: *Naive Bayesian*, *Support Vector Machine* and *Decision Tree*. In particular, *Support Vector Machine*, the best and most stable classifier, achieved a precision of 87% in this task.

In the next section, we detail how we analyze and extract versioning information patterns from the ontology URIs to select candidate ontology pairs. In Section 3, we describe the set of features which characterize ontology version pairs and the use of machine learning techniques to classify them. In Section 4, we discuss our evaluation using the three different classifier models: *Naive Bayesian* (NB), *Support Vector Machine* (SVM) and *Decision Tree* (DT). In Section 5, we discuss the relevant related work. Finally, in Section 6, we summarize the key contribution of this work and outline our plans for future work.

2 Finding Candidate Versioning Links Based on Ontology URIs

Detecting ontology version links raises the issue of *selecting pairs of ontologies to check whether they are versions or not*. In the context of SWSEs where there are thousands of collected ontologies, it would be unrealistic to apply complex comparisons on all the possible pairs of ontologies. Therefore, we need to define a straightforward method to discover candidate pairs of ontologies for such a comparison. In an initial work [3], we manually analyzed a representative sample (nearly 1000) of URIs from the ontology repository of the Watson SWSE. We noticed that many of them contain numerical information concerning the version of the ontology (following versioning patterns). Specifically, we identified three classes of such patterns: (1) where the versioning information is encoded in a single number; (2) where the versioning information is expressed by two numbers, which are either the month and year of a date or the two numbers of a major and minor release; and (3) where the versioning information is expressed by three numbers, which always correspond to a complete date. Based on these patterns, we designed six rules, specifically two rules for each pattern, to use to compare URIs which reflect the main characteristics of the classes:

R1 corresponds to the most straightforward case where there is only one numerical difference between two URIs. For example:

```
http://www.vistology.com/ont/tests/student1.owl;
http://www.vistology.com/ont/tests/student2.owl;
```

However, there can be many variants of such a pattern. In the following example, a time-stamp is used to mark a particular version of the ontology:

```
http://160.45.117.10/semweb/webrdf/#generate_time
   stamp_1176978024.owl
http://160.45.117.10/semweb/webrdf/#generate_time
   stamp_1178119183.owl
```

R2 corresponds to those cases where two numbers differ from one URI to the other. The version information corresponds to a version number, in which case the number on the left is more significant. For example:

```
http://lsdis.cs.uga.edu/projects/semdis/sweto/test
   bed_v1_0.owl
http://lsdis.cs.uga.edu/projects/semdis/sweto/test
   bed_v1_1.owl
```

R3 R4 correspond to those cases where two numbers differ from one URI to the other. The version information corresponds to a date including the year and month only, in which case, the year is more significant. For example:

```
http://loki.cae.drexel.edu/~wbs/ontology/2003/02/
   iso-metadata
http://loki.cae.drexel.edu/~wbs/ontology/2003/10/
   iso-metadata
http://loki.cae.drexel.edu/~wbs/ontology/2004/01/
   iso-metadata
http://loki.cae.drexel.edu/~wbs/ontology/2004/04/
   iso-metadata
```

R5 R6 correspond to the cases where three numerical differences exist between the considered URIs. In our dataset, we have not encountered examples other than the representation of dates using 3 numbers. Therefore, we only define the rules corresponding to dates, either in big endian or in little endian form. For example:

```
http://ontobroker.semanticweb.org/ontologies/ka2-
   onto-2000-11-07.daml
http://ontobroker.semanticweb.org/ontologies/ka2-
   onto-2001-03-04.daml
```

Moreover, our analysis also showed that these six rules cover nearly 90% of the versioning information codified in the URIs (we counted the number of ontology URIs detected for each rule out of 7000). The other 10% concern either versioning information expressed through the combination of words and numbers, e.g. `/january2007/` and `/october2006/` or patterns based on more than four numbers. Based on the three classes of versioning patterns, we devised a polynomial algorithm which compares URIs to extract the numerical differences and use the six rules to generate candidate versioning relations between ontologies [3]. Running it over a dataset of 7000 OWL/RDF-ontologies, the algorithm detected 24644 pairs of candidate ontology versions, in just a few minutes (between 3-4 minutes) on a MacBookPro laptop, Intel Core 2 Duo - 2.6 GHz. A subset of them (531 out of 24644) were manually analyzed, looking in particular at their content. The analysis showed that only about half of these pairs indicated versioning relations. In addition, we also noticed that when two ontologies were linked by versioning relations, it was possible to establish an overlapping between their vocabularies and sets of axioms. In the next Section we show how we exploit such features in a machine learning approach to automatically decide whether ontology pairs are in a versioning relation.

3 Applying Machine Learning Classifiers

Once we obtain an initial set of candidate ontology pairs which potentially represent versions, the issue is then to *automatically differentiate the pairs which are versions from the ones which are not.* To this purpose, we first identified a set of features which characterize ontology versions and then we applied machine learning techniques to classify the pairs in two classes: *PrevVersion* and *NotPrevVersion*.

3.1 Attributes

Here we describe in detail the set of ontology features that are exploited by Machine Learning techniques to identify versioning relations.

Length of Chain. The *Length of Chain* attribute represents the length of the sequence of successive ontologies that a pair is part of. We define and compute such *chains* of ontologies as the connected paths in the graph formed by the links detected by the mechanism described in Section 2. More formally,

Definition 1 (Ontology Chain). *Given a collection of pairs of ontologies* $P=\{(O_i, O_j)\}$*, an ontology chain is defined as the longest sequence of ontologies,* $O_1, O_2, ..., O_{n-1}, O_n$ *such that* $(O_k, O_{k+1}) \in P$ *and* O_k *represents the candidate previous version of* O_{k+1}*. Here, the collection* P *corresponds to the set of candidate pairs selected according to the approach described in Section 2. It is important to notice that, according to this approach, a given candidate pair can only be part of one chain of ontologies.*

Based on Definition [1], we computed 1365 ontology chains out of 24644 ontology pairs detected with the previous step. Manually analyzing a small number of these chains (39 out of 1365) we noticed that shorter chains (from 4 to 6) are more likely to actually represent sequences of ontology versions. The main reason for this is that many sequences come from automatically generated ontologies and URIs in which numbers are not representing version information but a "record number". This typically happens when database tuples are exported in RDF or OWL. In other terms, to each record from the database is assigned a number expressed in the URI when translated it into ontology language (e.g *...student-record-1*, *student-record-2*).

Similarity Measures. Similarity measures are considered to be relevant characteristics in studying the versioning problem [15], as a reasonable intuition is that ontology versions should be similar to a certain extent.

Ontology similarity has been described as a measure to assess how close two ontologies are [8]. Various ways to compute the similarity between two ontologies have been studied to be relevant in different application contexts [8,10,24]. However, no particular similarity function has been identified as being the most effective one for detecting versions of ontologies. Following the basic operations in ontology evolution of adding, deleting and modifying ontology axioms or just changing the ontology terminology [12,21], we focus here on two different similarity functions: 1) *lexicographicSimilarity* and 2) *syntacticSimilarity*, which are defined as follows

$$\textbf{\textit{lexicographicSimilarity(}}O_i,\ O_j\textbf{\textit{)}} = \frac{|Voc(O_j) \bigcap Voc(O_j)|}{max(|Voc(O_i)|, |Voc(O_j)|)}$$

where *Voc(*O_k*)* is the normalized[4] vocabulary VOC of the ontology O_k, k=i,j.

$$\textbf{\textit{syntacticSimilarity(}}O_i,O_j\textbf{\textit{)}} = \frac{|Axioms(O_i) \bigcap Axioms(O_j)|}{max(|Axioms(O_i)|, |Axioms(O_j)|)}$$

where *Axioms(*O_k*)* is the set of normalized[5] axioms of the ontology O_k, k=i,j.

Both these measures correspond to the ratio between the size of the overlap and the size of the largest vocabulary and set of axioms of the ontologies, respectively. We use the size of the larger ontology rather than the smaller because we believe that it gives intuitively better results in those cases where we need to compare ontologies of very different size.

Looking at Fig. [1], we notice that there is a non-trivial relationship between similarity and the presence of a versioning link in a given pair of ontologies. In other words, no threshold naturally emerges from these values that can help selecting pairs of ontologies not representing versions. Indeed, both very dissimilar and very similar ontologies appear unlikely to represent ontology versions. One

[4] We normalize the elements of the vocabulary of an ontology transforming upper case letters to lower case and removing separators such as underscore and hyphen.

[5] Normalized axioms are axioms where the elements of the vocabulary have been normalized.

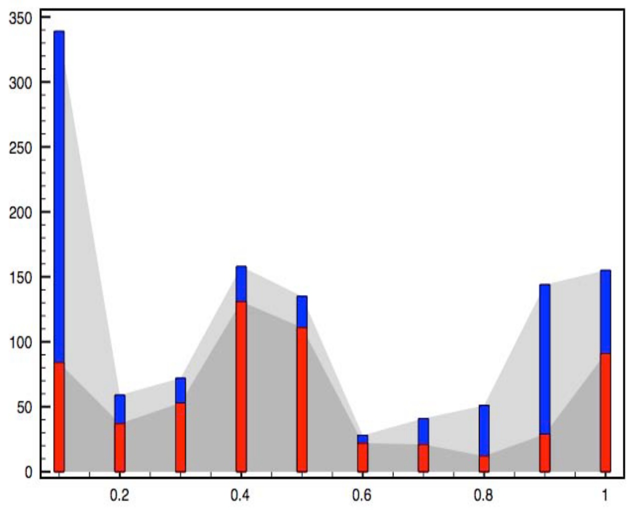


Fig. 1. Distribution of syntactic similarity in manually evaluated ontology pairs. Red represents pairs that are recognized as ontology versions, and blue represents the ones that are not.

of the goals of using machine learning classifiers here is therefore to find a mechanism to exploit this non-trivial relationship between similarity and versioning, combined with other characteristics of the ontology pairs.

Ontology Versioning Patterns. Based on the analysis of ontology URIs, we observed that some of the patterns described in Section 2 give better results than others in terms of both precision and recall. For example, R2, which represents patterns with two numbers (expressing a version number such as v1.1, v1.2), gives results which, while numerous, appear more accurate than R1, which considers only one number to express versioning information. Using such information, the classifier can compute different levels of confidence for each rule, and learn that some patterns provide more accurate information than others.

3.2 Methodology

A classifier model is an arbitrarily complex mapping from all-but-one dataset attributes to a class attribute C [5]. The specific form and creation of this mapping, or model, differs from classifier to classifier. In our case, the input attributes correspond to the three features described above and the class attribute, C, takes values in $\{PrevVersion, NotPrevVersion\}$. *PrevVersion* represents the class of ontology pairs (O_i, O_j) where O_i is a direct or indirect previous version of O_j, while *NotPrevVersion* represents the class of ontology pairs, (O_i, O_j), where O_i is not a previous version of O_j, directly or indirectly. In our experiments, we compared the performances of three different types of classifiers: *Naive Bayesian (NB), Support Vector Machine (SVM) and Decision Tree (DT)*, and analyzed to what extent the set of the proposed attributes supports the discovery of different versions of the same ontology. For our experiment, we use the Machine Learning Toolkit Weka [5] and the performance measures obtained are based on Weka's model implementation in its default configuration. The reader interested in finding out more details about the classifiers is referred to [11,17] for NB, [30] for SVM and [18] for DT.

A systematic approach to applying machine learning classifiers follows four phases: 1) *Training*; 2) *Testing*; 3) *Validation*; and 4) *Classification* [9,5]. Each phase receives an input set of *training*, *testing*, *validation* and *classification* tuples of feature values, respectively. In our case, an arbitrary tuple characterizing a candidate pair of ontologies (O_i, O_j) has the following structure: $[CL$, Rn, $VocSim$, $SynSim$, $C]$ where CL is the length of the chain that includes (O_i, O_j); Rn is the number of the rule (R1-R6) used to select (O_i, O_j); $VocSim$ is the *lexicographicSimilarity* measure value between O_i and O_j, $SynSim$ is the *syntacticSimilarity* measure value between O_i and O_j and the value of C is either *PrevVersion* or *NotPrevVersion*.

Given a classifier T, the *Training* phase is responsible for building the classifier model M_T. To do this, T receives the input set of *training* tuples, manually classified. The *Testing* phase is responsible for checking the accuracy of the model M_T. Here T receives the input set of *testing* tuples (disjoint from the *training* set), manually classified, and assigns a value to the class attribute C of each tuple. The accuracy of the model is measured by comparing the value of the class attribute C of the *testing* tuple set to the value assigned by the classifier T. The accuracy rate is given by the percentage of testing set samples correctly classified by the model. The *Validation* phase is in charge of evaluating the stability of the classifier. The classifier T receives the input set of manually classified *validation* tuples (disjoint from the *training* and *testing* sets). Running the model M_T over the *validation* set, the classifier computes its accuracy rate and performance one more time. Comparing these values with the ones obtained from the *Testing* phase, the stability of the classifier T is calculated. Finally, in the *Classification* phase T receives the input set of *classification* tuples and assigns one of two values $\{$*PrevVersion*, *NotPrevVersion*$\}$ to the class attribute C.

4 Evaluation

In this Section, we detail our evaluation of the approach described above, using the collection of OWL ontologies from the Watson SWSE.

4.1 Experimental Set-Up: Generating Datasets

Fig. 2 shows the main steps to build datasets for the classifiers. In particular, (Fig. 2 (a)) represents the Watson collection of 7000 ontologies. We first ran the versioning patterns based mechanism described in Section 2, which identified a set of 24644 candidate pairs of ontologies (Fig. 2 (b)). Then we manually analyzed a subset of these candidate pairs, producing 591 pairs as *PrevVersion*

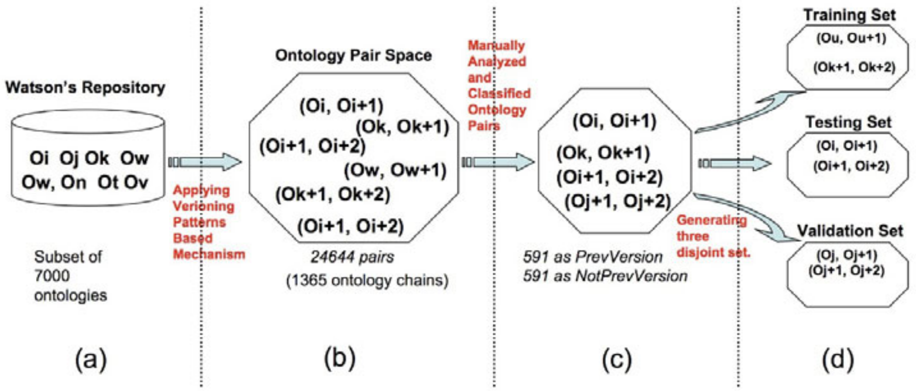


Fig. 2. Flow of building the datasets

and 591 pairs as *NotPrevVersion* (Fig. 2 (c)). From this set of 1182 pairs, we then created three disjoint datasets - a) *training* b) *testing* and c) *validation* (Fig 2 (d)), by means of the following steps:

1. We established a random order for the selected ontology pairs, using a simple algorithm to add a random number to the ontology pair (Fig. 2(c));
2. We sorted the set generated in the previous step;
3. We partitioned it into three equal sets, containing the same number of *PrevVersion* and *NotPrevVersion* pairs, obtaining the following dataset: 1) *training*; 2) *testing* and 3) *validation*, (Fig. 2 (d));
4. For each dataset, we generated Weka Attribute-Relation File Format (ARFF)[6] files of tuples.

4.2 Results of the Classifier's Performances

The bar diagrams in Fig. 3, Fig 4 and Fig. 5 show the performance results of the three classifiers, with respect to the rate of classification, precision and recall. As legend we use:

Classified(Correct) indicates the number of tuples correctly classified.

Classified(Incorrect) indicates the number of tuples incorrectly classified.

Precision(PrevVersion) indicates the proportion of tuples correctly classified as $PrevVersion$ among all those that were classified as $PrevVersion$, i.e., $\frac{CPV \cap EPV}{CPV}$ where CPV is the set of tuples classified as $PrevVersion$, and EPV is the set of tuples identified as $PrevVersion$ through manual evaluation.

[6] An ARFF file is a text file that describes a list of instances sharing a set of attributes, see `http://www.cs.waikato.ac.nz/ml/weka/`.

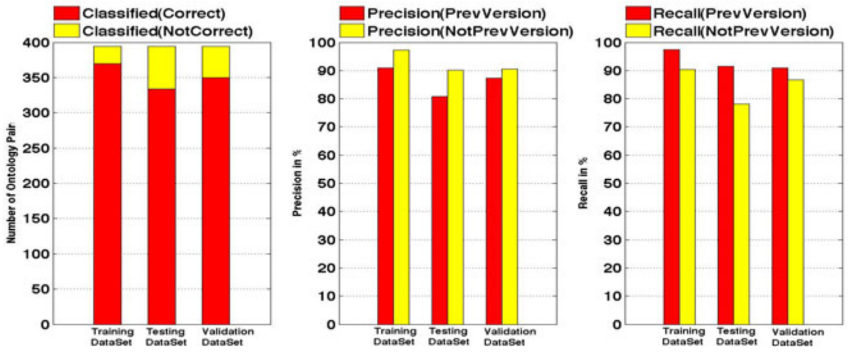


Fig. 3. *Left*: DT classification performance; *Middle*: DT precision performance; *Right*: DT recall performance

Precision(NotPrevVersion) similarly, indicates the proportion of tuples correctly classified as $NotPrevVersion$ among all those that where classified as $NotPrevVersion$

Recall(PrevVersion) indicates the proportion of tuples classified as $Prev Version$, among those that were identified as $PrevVersion$ in our evaluation, i.e., $\frac{CPV \cap EPV}{EPV}$.

Recall(NotPrevVersion) similarly, indicates the proportion of tuples classified as $NotPrevVersion$, among those that were identified as $NotPrev Version$ in our evaluation.

In general, all three classifiers performed well. SVM performed better than the other two reaching an average of 90% of the tuples correctly classified, an average precision of 87.2% and an average recall of 95% of tuples classified in the *PrevVersion* class[7]. Moreover, SVM is also more stable than DT and NB over the three phases of *Training*, *Testing* and *Validation*. In fact, compared to the other two classifiers, SVM presents very little fluctuations of the values of *Classification(Correct)*, *Precision (PrevVersion)* and *Recall (PrevVersion)* (Fig. 4). This confirms the fact that SVM is, in general, more accurate and gives better results than NB and DT [30]; in particular with complex distribution of data such as the one shown in Fig. 1, where the relation between the data attribute and the class attribute is complex. Furthermore, while having similar complexity to NB (polynomial time, [30,17]), SVM was also the fastest. It took less than one minute to build the model and to use it to classify new tuples. DT is the second best classifier with an average of 89% of tuples correctly classified, an average precision of 86% and an average recall of 93% of tuples classified as *PrevVersion*. It was less stable and slower in running performance than NB with an average of 80%

[7] The precision and recall measures being correlated for $NotPrevVersion$, we focus here on the values for $PrevVersion$. It is worth noticing also that the measures of recall are relative to the set of ontology pairs selected using the method described in Section 2.

of tuples correctly classified, an average precision of 71.5% and an average recall of 100% of tuples classified as *PrevVersion*. This can be observed comparing the diagrams in the middle of Fig. 3 and Fig. 5. The DT *Precision(PrevVersion)* performance is more fluctuant than the one of NB. Similar differences appear for *Classification(Correct)* and *Recall(PrevVersion)*. Moreover, due to the exponential complexity of DT [16], NB required about half of the time of DT (three minutes) to built the model and to use it. Finally, NB was the worst performer with 71% classification accuracy compared to 87.2% for SVM. This may be due to the fact that NB is based on a very strong assumption that the ontology versioning features are all independent from each other. In our case, however, *syntacticSimilarity* and *lexicographicSimilarity* are related to each other.

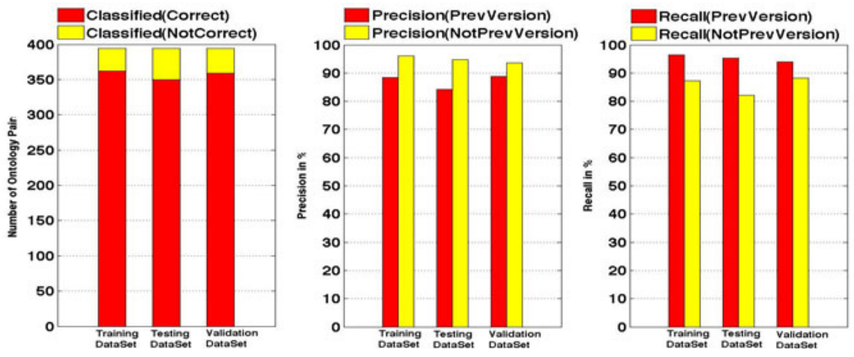


Fig. 4. *Left*: SVM classification performance; *Middle*: SVM precision performance; *Right*: SVM recall performance.

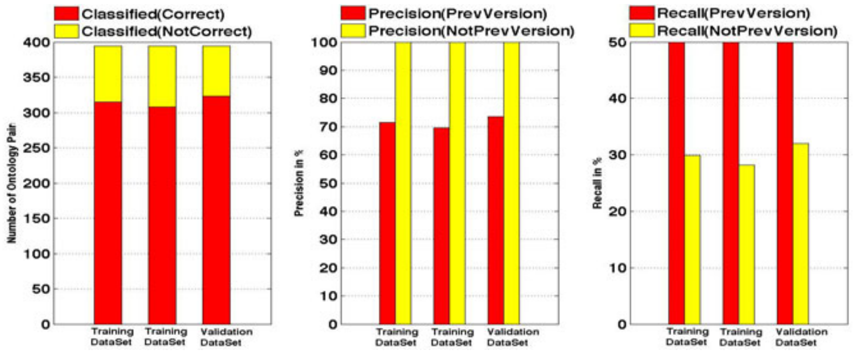


Fig. 5. *Left*: NB classification performance; *Middle*: NB precision performance; *Right*: NB recall performance.

4.3 Discussion

One of the advantages of DT over SVM was that using DT we could analyze the tree model it generates to examine in details the effect of each attribute on

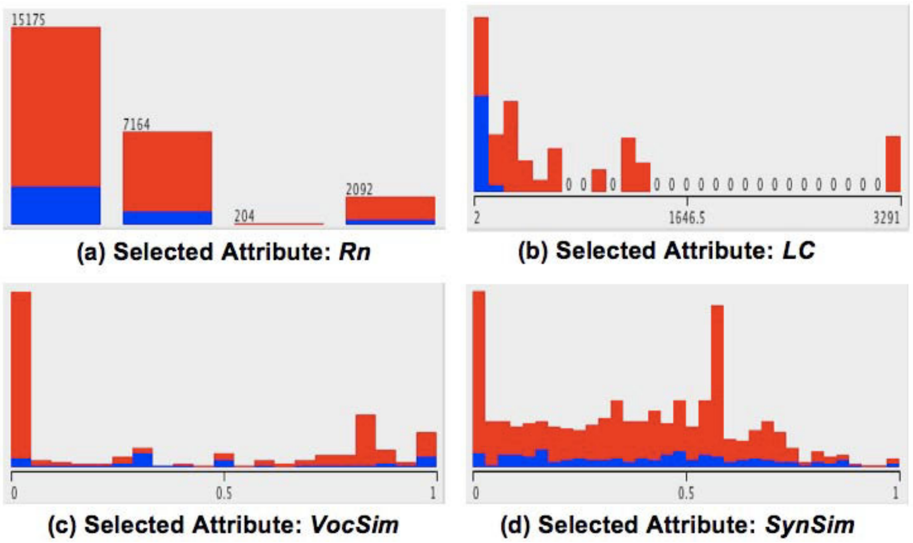


Fig. 6. Each diagram represents the single attribute contribution to identifying ontology versions. Blue color represents pairs classified as versions by the Decision Tree classifier, and red color are pairs which are not recognized as versions.

the identification of ontology versions (see Fig. 6). The ontology URI versioning patterns (R1-R6 rules) have shown to represent a good starting point to detect candidate versioning pairs of ontologies to be analyzed further by the classifiers, Fig. 6 (a).

Unsurprisingly, we could observe that the length of the ontology chains (LC) is the most relevant attribute to distinguish ontology versions from otherwise related ontology pairs Fig. 6 (b). Basically, we can easily show in our results that long sequences of versions tend to be rare, and that most of the actual version sequences have no more than 6 steps. This can probably be explained by the fact that publishing ontologies online is a relatively recent practice. The the average length of the chains of ontology versions online can therefore be expected to increase with time. As "secondary attributes", the similarity measures we employed also played an important role in the classification, particularly in distinguishing version pairs in the cases where the length of the chain was not sufficient (i.e., when the chains are reasonably short, see Fig. 6 (c,d)). Looking at Fig. 7 obtained from plotting the classification of our dataset with respect to both the lexicographic and syntactic similarity measures, an interesting phenomenon appears. Indeed, it shows that the place on the plot where one would most likely find versions is not at any of the extremes, but somewhere in the middle. This seems to indicate that, contrary to our initial intuition, ontology version pairs not only have a certain level of overlapping, but also a significant number of "changes", both in their vocabularies and structures.

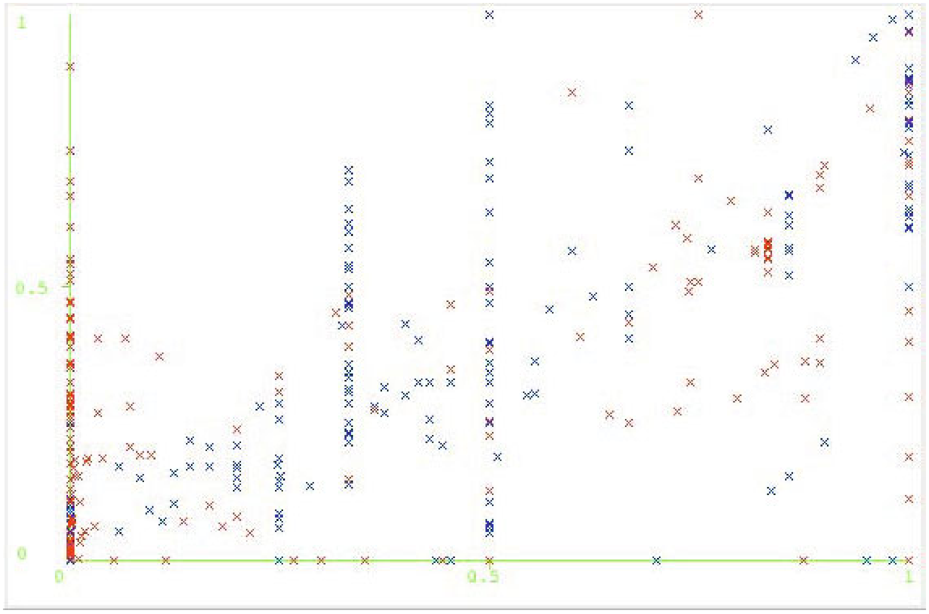


Fig. 7. Plot of the syntactic (x) and the lexicographic (y) similarity measures. Blue dots represent pairs classified as versions by the Decision Tree classifier, and red dots are pairs which are not recognized as versions.

5 Related Work

To the best of our knowledge, detecting versions of ontologies in the context of SWSE systems has not yet been approached by any of the current SWSEs, such as Swoogle, Sindice and Watson.

From both theoretical and practical points of view, most of the previous works address the ontology versioning problem focusing on *Ontology Version management* [20,22,27,28] and on *Ontology Evolution* [12,21,23,25] during the phase of ontology development. Klein [19] defines the *Ontology Versioning* problem as "*the ability to manage ontology changes and their effects by creating and maintaining different variants/mutants/versions of the ontology*". However, such sophisticated versioning mechanisms or system are not yet available. Heflin [15] formalized a semantic model for managing ontologies in distributed environments, such as the Web. On this basis, Heflin [14] developed Simple HTML Ontology Extensions (SHOE) as an extension of HTML to represent ontology-based knowledge using additional tags such as Backward-Compatible-With to specify that the current version of an ontology is compatible with previous versions. Later, Patel-Schneider [4] extended OWL (Ontology Web Language) providing new mechanisms, such as *owl:versionInfo*, *owl:priorVersion*, *owl:backwardCompatibleWith* and *owl:incompatibleWith*, to make the links explicit between versions of ontologies. The main drawback of all these mechanisms is that they are very rarely used, as ontology developers hardly ever make the effort of applying them. Instead, they tend to codify information related to the version of an ontology directly in its URI [3]. Hartmann and Palma proposed the Ontology Metadata Vocabulary (OMV) [13], which provides an extensive vocabulary for describing ontologies, including provenance (e.g., creator, contributor); relationship (e.g., import, backward compatibility), format (e.g., ontology language and syntax), etc. However, the ontologies collected by SWSEs are rarely described using OMV. Several practical approaches have focused on comparing two different versions of ontologies in order to find out the differences and identify the changes. In particular, PROMTDIFF [26] compares the structure of ontologies and OWLDiff (`http://semanticweb.org/wiki/OWLDiff`) computes the differences by checking the entailment of the two sets of axioms. SemVersion [31] compares two ontologies and computes the differences at both structural and semantic levels. OntoDiff is a software tool that compares versions to identify and manage the ontology changes locally at both structural and content levels [29].

Closer to our work but applied on XML documents, [9] attempts to identify versions of documents using naive Bayes classifiers. The authors use a similarity measure dedicated to XML documents as input for the classifiers, and apply the approach on a set of automatically generated documents in a closed domain.

6 Conclusion and Future Work

In this paper, we have presented an approach to detect versions of ontologies in the context of Semantic Web Search Engines. The method is based on two main steps. The first step tackles the issue of selecting pairs of ontologies as candidate versions (see Section 2). The second step deals with the issue of deciding whether the selected ontology pairs are versions or not (see Section 3). Our approach solution is based on the application of machine learning techniques to a set of attributes, which tend to characterize ontology versions. In particular, we compared the *Naive Bayesian*, *Support Vector Machine* and *Decision Tree* classifiers. SVM achieved the highest level of precision in this task, 87%. This result shows that characteristics such as similarity and length of chain can be used as features to decide whether a pair of ontologies represent a version link or not, by using complex algorithms from Machine Learning. Our approach therefore provides a reasonably accurate method that can be used by a SWSE to detect ontology versions automatically.

This work will be the starting point for both empirical and practical future studies. From an empirical point of view, it is possible to analyze different ontology versions to better understand the ontology engineering practices behind the modeling process. These different versions provide data about the development process of ontologies, showing how they reach stability or adapt to changes in the domain. In other words, once we have detected the links between different versions of the ontologies, it becomes possible to explore how ontology versions actually evolve on the Semantic Web, in particular with the aim of discovering relevant *patterns* in ontology evolution. It will also be interesting to study, for example, the inter-consistency of ontology versions. Furthermore, as part of our broader work on building a framework for the management of ontology relationships (see e.g., [2]), one of our future directions of research is to consider versioning links in combination with other high level ontology relationships such as *inclusion*, *compatibility*, or *agreement* [6].

From a practical point of view, we believe that making explicit such relations between ontologies, including versioning links, can facilitate the ontology selection in a SWSE. This assumption is currently being tested based on an integration with the Watson system (see `http://smartproducts1.kmi.open.ac.uk:8080/WatsonWUI-K/`).

References

1. Alani, H., Brewster, C., Shadbolt, N.: Ranking ontologies with AKTiveRank. In: 5th Int. Semantic Web Conf., Athens, Georgia, USA (2006)
2. Allocca, C., d'Acquin, M., Motta, E.: Door: Towards a formalization of ontology relations. In: Proc. of the IC on Knowledge Eng. and Ontology Dev., KEOD (2009)
3. Allocca, C., d'Aquin, M., Motta, E.: Detecting different versions of ontologies in large ontology repositories. In: International Workshop on Ontology Dynamic at International Semantic Web Conference (2009)
4. Bechhofer, S., Harmelen, F.V., Hendler, J., Horrocks, I., McGuinness, D.L., Patel-Schneider, P.F., Stein, L.A.: OWL Web Ontology Language Reference. Technical report (2004)
5. Bouckaert, R.R., Frank, E., Hall, M., Kirkby, R., Reutemann, P., Seewald, A., Scuse, D.: WEKA Manual for Version 3-6-2, Berlin (2010)
6. d'Aquin, M.: Formally measuring agreement and disagreement in ontologies. In: K-CAP, pp. 145–152. ACM, New York (2009)
7. d'Aquin, M., Motta, E., Sabou, M., Angeletou, S., Gridinoc, L., Lopez, V., Guidi, D.: Toward a new generation of semantic web applications. Intelligent Systems 23(3), 20–28 (2008)
8. David, J., Euzenat, J.: Comparison between ontology distances (Preliminary results). In: Sheth, A.P., Staab, S., Dean, M., Paolucci, M., Maynard, D., Finin, T., Thirunarayan, K. (eds.) ISWC 2008. LNCS, vol. 5318, pp. 245–260. Springer, Heidelberg (2008)
9. de Brum Saccol, D., Edelweiss, N., de Matos Galante, R., Zaniolo, C.: Xml version detection. In: ACM Symp. on Doc. Eng., pp. 79–88 (2007)
10. Ehrig, M., Haase, P., Hefke, M., Stojanovic, N.: Similarity for ontologies - a comprehensive framework. In: Karagiannis, D., Reimer, U. (eds.) PAKM 2004. LNCS (LNAI), vol. 3336. Springer, Heidelberg (2004)
11. Friedman, N., Geiger, D., Goldszmidt, M., Provan, G., Langley, P., Smyth, P.: Bayesian network classifiers. Machine Learning, 131–163 (1997)
12. Antoniou, G., Flouris, G., Plexousakis, D.: Evolving ontology evolution. In: Wiedermann, J., Tel, G., Pokorný, J., Bieliková, M., Štuller, J. (eds.) SOFSEM 2006. LNCS, vol. 3831, pp. 14–29. Springer, Heidelberg (2006)
13. Hartmann, J., Palma, R., et al.: Ontology metadata vocabulary and applications. pp. 906–915 (October 2005)
14. Heflin, J., Hendler, J.A.: Dynamic ontologies on the web. In: Proc. of the 7th Nat. Conf. on AI and 12th Conf. on Inn. App. of AI, pp. 443–449. AAAI Press / The MIT Press (2000)

15. Heflin, J., Pan, Z.: A model theoretic semantics for ontology versioning. In: McIlraith, S.A., Plexousakis, D., van Harmelen, F. (eds.) ISWC 2004. LNCS, vol. 3298, pp. 62–76. Springer, Heidelberg (2004)
16. Hyafil, L., Rivest, R.: Constructing optimal binary decision trees is np-complete. Information Processing Letters 5(1) (1976)
17. John, G., Langley, P.: Estimating continuous distributions in bayesian classifiers. In: Proc. of the 11th Conf. on Uncertainty in AI, pp. 338–345. Morgan Kaufmann, San Francisco (1995)
18. Quinlan, J.R.: Induction of decision trees. Machine Learning 1, 81–106 (1986)
19. Klein, M., Fensel, D.: Ontology versioning on the semantic web. In: Proc. of the Inter. Semantic Web Working Symposium (SWWS), pp. 75–91 (2001)
20. Klein, M., Kiryakov, A., Ognyanoff, D., Fensel, D.: Finding and specifying relations between ontology versions (2002)
21. Liang, Y., Alani, H., Shadbolt, N.R.: Change management: The core task of ontology versioning and evolution. In: Proc. of Postgraduate Research Conf. in Elect, Photonics, Communication. and Net, and Computing Science 2005 Lancaster, United Kingdom (PREP 2005), pp. 221–222 (2005)
22. Maedche, A., Alexander, Motik, B., Stojanovic, L., Studer, R., Volz, R.: Managing multiple ontologies and ontology evolution in ontologging. In: Proc of the IFIP 17th World Computer Congress - TC12 Stream on IIP, Deventer, The Netherlands, pp. 51–63. Kluwer, B.V., Dordrecht (2002)
23. Maedche, A., Motik, B., Stojanovic, L., Stojanovic, N.: User-driven ontology evolution management, pp. 285–300. Springer, Heidelberg (2002)
24. Maedche, A., Staab, S.: Comparing ontologies-similarity measures and a comparison study. In: Proc. of EKAW 2002 (2002)
25. Noy, N.F., Klein, M.: Ontology evolution: Not the same as schema evolution. Knowledge and Information Systems 6(4), 428–440 (2004)
26. Noy, N.F., Musen, M.A.: Promptdiff: A fixed-point algorithm for comparing ontology versions. In: 18th National Conf. on Artificial Intelligence, AAAI (2002)
27. Noy, N.F., Musen, M.A.: Ontology versioning in an ontology management framework. IEEE Intelligent Systems 19, 6–13 (2004)
28. Redmond, T., Smith, M., Drummond, N., Tudorache, T.: Managing change: An ontology version control system. In: OWLED. CEUR Proc., vol. 432 (2008)
29. Tury, M., Bieliková, M.: An approach to detection ontology changes. In: ICWE Proc. of the 6th Int. Conf. on Web Eng., p. 14. ACM, New York (2006)
30. Vapnik, V.: The Nature of Statistical Learning Theory. Springer, New York (1995)
31. Volkel, M.: D2.3.3.v2 SemVersion Versioning RDF and Ontologies. EU-IST Network of Excellence (NoE) IST-2004-507482 KWEB

A Tableaux-Based Algorithm for $\mathcal{SHIQ}$ with Transitive Closure of Roles in Concept and Role Inclusion Axioms

Chan Le Duc[1], Myriam Lamolle[1], and Olivier Curé[2]

[1] LIASD Université Paris 8 - IUT de Montreuil, France
{chan.leduc,myriam.lamolle}@iut.univ-paris8.fr
[2] LIGM Université Paris-Est, France
ocure@univ-mlv.fr

Abstract. In this paper, we investigate an extension of the description logic $\mathcal{SHIQ}$—a knowledge representation formalism used for the Semantic Web—with transitive closure of roles occurring not only in concept inclusion axioms but also in role inclusion axioms. It was proved that adding transitive closure of roles to $\mathcal{SHIQ}$ without restriction on role hierarchies may lead to undecidability. We have identified a kind of role inclusion axioms that is responsible for this undecidability and we propose a restriction on these axioms to obtain decidability. Next, we present a tableaux-based algorithm that decides satisfiability of concepts in the new logic.

1 Introduction

The ontology language OWL-DL [1] is widely used to formalize semantic resources on the Semantic Web. This language is mainly based on the description logic $\mathcal{SHOIN}$ which is known to be decidable [2]. Although $\mathcal{SHOIN}$ is expressive and provides *transitive roles* to model transitivity of relations, we can find several applications in which *the transitive closure of roles*, that is more expressive than transitive roles, is necessary. The difference between transitive roles and the transitive closure of roles is clearer when they are involved in role inclusion axioms. For instance, if we denote by R^- and R^+ the inverse and transitive closure of a role R respectively then it is obvious that the concept $\exists \mathsf{R}^+.(\mathsf{C} \sqcap \forall \mathsf{R}^-.\bot)$ is unsatisfiable w.r.t. an empty TBox and the trivial axiom $\mathsf{R} \sqsubseteq \mathsf{R}^+$. If we now substitute R^+ for a transitive role R_t such that $\mathsf{R} \sqsubseteq \mathsf{R}_t$ (i.e. we substitute each occurrence of R^+ in axioms and concepts for R_t) then the concept $\exists \mathsf{R}_t.(\mathsf{C} \sqcap \forall \mathsf{R}^-.\bot)$ becomes satisfiable. The point is that an instance of R^+ represents a sequence of instances of R but an instance of R_t corresponds to a sequence of instances of *itself*.

In several applications, we need to model successive events and relationships between them. An event is something oriented in time, i.e. we can talk about endpoints of an event, or a chronological order of events. When an event of some kind occurs it can trigger an event (or a sequence of events) of another kind. In this situation, it may be suitable to use a role to model an event. If we denote roles event and event' for two kinds of events then the axiom $(\mathsf{event} \sqsubseteq \mathsf{event}')$ expresses the fact that when an event

© Springer-Verlag Berlin Heidelberg 2011

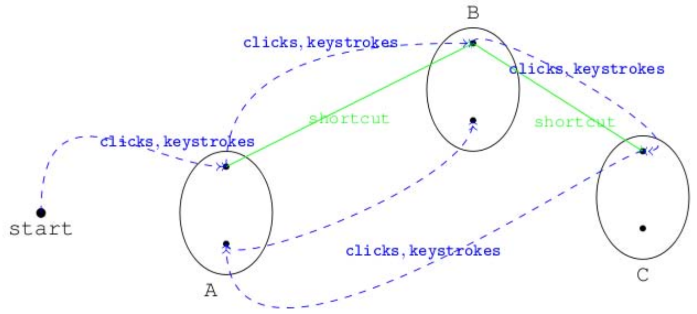


Fig. 1. Mouse clicks, keystrokes and shortcuts

of the first kind occurs it implies one event or a sequence of events of the second kind. To express "a sequence of events" we can define event' to be transitive. However, the semantics of transitive roles is not sufficient to describe this behaviour since the transitive role event' can represent a sequence of itself but not a sequence of another role. Such behaviours can be found in the following example.

Example 1. Let $\mathbf{S}$ be the set of all states of applications running on a computer. We denote by $\mathsf{A}, \mathsf{B}, \mathsf{C} \subseteq \mathbf{S}$ the sets of states of applications A, B, C, respectively. A user can perform a mouse-click or keystroke to change states. She can type a shortcut (combination of keys) to go from A to B or from B to C. This action corresponds to a sequence of mouse-clicks or keystrokes. The system's behaviour is depicted in Figure 1. In such a system, users may be interested in the following question: "from the application A, can one go through the application B to get directly to the application C by a mouse-click or keystroke ?".

We now use a description logic with transitive closure of roles to express the constraints as described above. To do this, we use a role next to model *mouse clicks* or *keystrokes* and a role jump to model *shortcuts* in the following axioms:

(i) $\mathsf{start} \sqsubseteq \neg\mathsf{A} \sqcap \neg\mathsf{B} \sqcap \neg\mathsf{C}$; $\mathsf{X} \sqcap \mathsf{Y} \sqsubseteq \bot$ with $\mathsf{X}, \mathsf{Y} \in \{\mathsf{A}, \mathsf{B}, \mathsf{C}\}$ and $\mathsf{X} \neq \mathsf{Y}$;
(ii) $\mathsf{A} \sqsubseteq \exists\mathsf{jump}.\mathsf{B}$; $\mathsf{A} \sqsubseteq \exists\mathsf{jump}.\mathsf{C}$; $\mathsf{B} \sqsubseteq \exists\mathsf{jump}.\mathsf{C}$;
(iii) $\mathsf{start} \sqsubseteq \forall\mathsf{next}^-.\bot$; $\mathsf{jump} \sqsubseteq \mathsf{next}^+$;

Under some operating systems, users cannot switch directly from an application to a particular one just by one mouse click or keystroke. We can express this constraint with the following axiom:

(iv) $\mathsf{C} \sqcap \exists\mathsf{next}^-.\mathsf{B} \sqsubseteq \bot$;

In this case, the concept $(\mathsf{A} \sqcap \exists\mathsf{next}^+.(\mathsf{C} \sqcap \exists\mathsf{next}^-.\mathsf{B}))$ capturing the question above is unsatisfiable w.r.t. the axioms presented.

Such examples motivate the study of Description Logics (DL) that allow for the transitive closure of roles to occur in both concept and role inclusion axioms. In this work, we introduce a DL that can model systems as described in Example 1 and propose a tableaux-based decision procedure for the concept satisfiability problem in this DL.

To the best of our knowledge, the decidability of $\mathcal{SHIQ}$ with transitive closure of roles, is unknown. [3] and [4] have established decision procedures for concept satisfiability in $\mathcal{SHI}_+$ ($\mathcal{SHI}$ with transitive closure of roles in concept and role inclusion axioms) and $\mathcal{SHIO}_+$ ($\mathcal{SHI}_+$ with nominals). These decision procedures have used *neighborhoods* for representing an individual with its neighbors in a model in order to build completion graphs. In the literature, many decidability results in DLs can be obtained from their counterparts in modal logics ([5], [6]). However, these counterparts do not take into account expressive role inclusion axioms. In particular, [6] has shown decidability of a very expressive DL, so-called $\mathcal{CATS}$, including $\mathcal{SHIQ}$ with the transitive closure of roles but not allowing it to occur in role inclusion axioms. [6] has pointed out that the complexity of concept subsumption in $\mathcal{CATS}$ is ExpTime-complete by translating $\mathcal{CATS}$ into the logic Converse PDL in which inference problems are well studied.

Recently, there have been some works in [7] and [8] which have attempted to augment the expressiveness of role inclusion axioms. A decidable logic, namely $\mathcal{SROIQ}$, resulting from these efforts allows for new role constructors such as composition, disjointness and negation. In addition, [9] has introduced a DL, so-called $\mathcal{ALCQI}b^+_{reg}$, which can capture $\mathcal{SRIQ}$ ($\mathcal{SROIQ}$ without nominal), and obtained the worst-case complexity (ExpTime-complete) of the satisfiability problem by using automata-based technique. $\mathcal{ALCQI}b^+_{reg}$ allows for a rich set of operators on roles by which one can simulate role inclusion axioms. However, transitive closures in role inclusion axioms are expressible neither in $\mathcal{SROIQ}$ nor in $\mathcal{ALCQI}b^+_{reg}$.

Tableaux-based algorithms for expressive DLs such as $\mathcal{SHIQ}$ [10] and $\mathcal{SHOIQ}$ [11] result in efficient implementations. This kind of algorithms relies on two structures, the so-called *tableau* and *completion graph*. Roughly speaking, a tableau for a concept represents a model for the concept and it is possibly infinite. A tableau translates satisfiability of all given concept and role inclusion axioms into the satisfiability of constraints imposed *locally* on each individual of the tableau by the semantics of concepts in the individual's label. This feature of tableaux will be called *local satisfiability property*. To check satisfiability of a concept, tableaux-based algorithms try to build a completion graph whose finiteness is ensured by a technique, the so-called *blocking technique*. It provides a termination condition and guarantees soundness and completeness. The underlying idea of the blocking mechanism is to detect "loops" which are repeated pieces of a completion graph. When transitive closure of roles is added to knowledge bases, this blocking technique allows us to lengthen paths through such loops in order to satisfy semantic constraints imposed by transitive closures. The algorithm in [12] for satisfiability in $\mathcal{ALC}_{reg}$ (including the transitive closure of roles and other role operators) introduced a method to deal with loops which can hide unsatisfiable nodes. This method detects on so-called *concept trees*, "good" or "bad" cycles that are similar to those between blocking and blocked nodes on completion trees.

To deal with transitive closure of roles occurring in terms such as $\exists Q^+.C$, we have to introduce a new expansion rule to build completion trees such that it can generate a path formed from nodes that are connected by edges whose label contains role Q. In addition, this rule propagates terms $\exists Q^+.C$ to each node along with the path before reaching a node whose label includes concept C. Such a path may go through blocked and blocking nodes and has an arbitrary length. To handle transitive closures of roles occurring in role inclusion axioms such as $R \sqsubseteq Q^+$, we use another new expansion rule that translates satisfaction of such axioms into satisfaction of a term $\exists Q^+.\Phi$. From the path generated from $\exists Q^+.\Phi$, a cycle can be formed to satisfy the semantic constraint imposed by $R \sqsubseteq Q^+$. Since the role Q, which will be defined to be *simple*, does not occur in number restrictions, the cycle obtained from this method does not violate other semantic constraints.

The contribution of the present paper consists of (i) designing a decidable logic, namely $\mathcal{SHIQ}_+$, with a new definition for simple roles and (ii) proposing a tableaux-based algorithm for satisfiability of concepts in $\mathcal{SHIQ}_+$.

2 The Description Logic $\mathcal{SHIQ}_+$

The logic $\mathcal{SHIQ}_+$ is an extension of $\mathcal{SHIQ}$ introduced in [11] by allowing transitive closure of roles to occur in concept and role inclusion axioms. In this section, we present the syntax and semantics of the logic $\mathcal{SHIQ}_+$. This includes an extension of the definition of simple roles to $\mathcal{SHIQ}_+$ and the definition of inference problems that we are interested in. The definitions reuse some notation introduced in [11].

Definition 1. *Let* $\mathbf{R}$ *be a non-empty set of* role names. *We denote* $\mathbf{R}_\mathsf{I} = \{P^- \mid P \in \mathbf{R}\}$ *and* $\mathbf{R}_+ = \{Q^+ \mid Q \in \mathbf{R} \cup \mathbf{R}_\mathsf{I}\}$.

$*$ *The set of* $\mathcal{SHIQ}_+$*-roles is* $\mathbf{R} \cup \mathbf{R}_\mathsf{I} \cup \mathbf{R}_+$. *A* role inclusion axiom *is of the form* $R \sqsubseteq S$ *for two* $\mathcal{SHIQ}_+$*-roles* R *and* S. *A* role hierarchy $\mathcal{R}$ *is a finite set of role inclusion axioms.*

$*$ *An interpretation* $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$ *consists of a non-empty set* $\Delta^\mathcal{I}$ *(*domain*) and a function* $\cdot^\mathcal{I}$ *which maps each role name to a subset of* $\Delta^\mathcal{I} \times \Delta^\mathcal{I}$ *such that, for* $R \in \mathbf{R}$, $Q^+ \in \mathbf{R}_+$,
$R^{-\mathcal{I}} = \{\langle x, y\rangle \in (\Delta^\mathcal{I})^2 \mid \langle y, x\rangle \in R^\mathcal{I}\}$, $(Q^+)^\mathcal{I} = \bigcup_{n>0} (Q^n)^\mathcal{I}$ *with* $(Q^1)^\mathcal{I} = Q^\mathcal{I}$ *and* $(Q^n)^\mathcal{I} = \{\langle x, y\rangle \in (\Delta^\mathcal{I})^2 \mid \exists z \in \Delta^\mathcal{I}, \langle x, z\rangle \in (Q^{n-1})^\mathcal{I}, \langle z, y\rangle \in Q^\mathcal{I}\}$.
An interpretation $\mathcal{I}$ *satisfies a role hierarchy* $\mathcal{R}$ *if* $R^\mathcal{I} \subseteq S^\mathcal{I}$ *for each* $R \sqsubseteq S \in \mathcal{R}$. *Such an interpretation is called a* model *of* $\mathcal{R}$, *denoted by* $\mathcal{I} \models \mathcal{R}$.

$*$ *To simplify notations for nested inverse roles and transitive closures of roles, we define two functions* $\cdot^\ominus$ *and* $\cdot^\oplus$ *as follows:*

$$R^\ominus = \begin{cases} R^- & \text{if } R \in \mathbf{R}; \\ S & \text{if } R = S^- \text{ and } S \in \mathbf{R}; \\ (S^-)^+ & \text{if } R = S^+,\ S \in \mathbf{R}, \\ S^+ & \text{if } R = (S^-)^+,\ S \in \mathbf{R} \end{cases} \quad R^\oplus = \begin{cases} R^+ & \text{if } R \in \mathbf{R}; \\ S^+ & \text{if } R = (S^+)^+ \text{ and } S \in \mathbf{R}; \\ (S^-)^+ & \text{if } R = S^- \text{ and } S \in \mathbf{R}; \\ (S^-)^+ & \text{if } R = (S^+)^- \text{ and } S \in \mathbf{R} \end{cases}$$

$*$ *A relation* $\sqsubseteq^*$ *is defined as the transitive-reflexive closure* $\mathcal{R}^+$ *of* $\sqsubseteq$ *on* $\mathcal{R} \cup \{R^\ominus \sqsubseteq S^\ominus \mid R \sqsubseteq S \in \mathcal{R}\} \cup \{R^\oplus \sqsubseteq S^\oplus \mid R \sqsubseteq S \in \mathcal{R}\} \cup \{Q \sqsubseteq Q^\oplus \mid Q \in \mathbf{R} \cup \mathbf{R}_\mathsf{I}\}$. *We denote* $S \equiv R$ *iff* $R \sqsubseteq^* S$ *and* $S \sqsubseteq^* R$.

$*$ *A role* R *is called* simple *w.r.t.* $\mathcal{R}$ *iff (i)* $Q^\oplus \sqsubseteq^* R \notin \mathcal{R}^+$ *for each* $Q \in \mathbf{R} \cup \mathbf{R}_\mathsf{I}$, *and (ii)* $R' \sqsubseteq^* R$, $P \sqsubseteq^* R'^\oplus \in \mathcal{R}^+$ *implies* $P \sqsubseteq^* R' \in \mathcal{R}^+$.

The reason for the introduction of two functions $\cdot^\ominus$ and $\cdot^\oplus$ in Definition 1 is that they avoid using R^{--} and R^{++}, moreover it remains a unique nested case $(R^-)^+$.

Notice that a transitive role S (i.e. $\langle x, y\rangle \in S^{\mathcal{I}}, \langle y, z\rangle \in S^{\mathcal{I}}$ implies $\langle x, z\rangle \in S^{\mathcal{I}}$ where $\mathcal{I}$ is an interpretation) can be expressed by using a role axiom $S^{\oplus} \sqsubseteq S$. In addition, a role R which is *simple* according to Definition 1 is simple according to [10] as well. In fact, if $Q^{\oplus} \mathrel{\underline{\boxast}} R \notin \mathcal{R}^{+}$ for each $Q \in \mathbf{R} \cup \mathbf{R}_{\mathsf{I}}$ then there is no transitive role S such that $S \mathrel{\underline{\boxast}} R \in \mathcal{R}^{+}$. Finally, if $R \mathrel{\underline{\boxast}} S \in \mathcal{R}^{+}$ and R is not simple according to Definition 1 then S is not simple according to Definition 1.

Definition 2. *Let* $\mathbf{C}$ *be a non-empty set of* concept names.

$*$ *The set of $\mathcal{SHIQ}_{+}$-concepts is inductively defined as the smallest set containing all C in* $\mathbf{C}$, $\top$, $C \sqcap D$, $C \sqcup D$, $\neg C$, $\exists R.C$, $\forall R.C$, $(\leq n\, S.C)$ *and* $(\geq n\, S.C)$ *where C and D are $\mathcal{SHIQ}_{+}$-concepts, R is an $\mathcal{SHIQ}_{+}$-role and S is a simple role. We denote $\bot$ for $\neg\top$.*

$*$ *An interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consists of a non-empty set $\Delta^{\mathcal{I}}$ (*domain*) and a function $\cdot^{\mathcal{I}}$ which maps each concept name to a subset of $\Delta^{\mathcal{I}}$ such that*
$\top^{\mathcal{I}} = \Delta^{\mathcal{I}},\ (C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}},\ (C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}},\ (\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \backslash C^{\mathcal{I}},$
$(\exists R.C)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid \exists y \in \Delta^{\mathcal{I}},\ \langle x, y\rangle \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\},$
$(\forall R.C)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid \forall y \in \Delta^{\mathcal{I}},\ \langle x, y\rangle \in R^{\mathcal{I}} \Rightarrow y \in C^{\mathcal{I}}\},$
$(\geq n\, S.C)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid \mathsf{card}\{y \in C^{\mathcal{I}} \mid \langle x, y\rangle \in S^{\mathcal{I}}\} \geq n\},$
$(\leq n\, S.C)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid \mathsf{card}\{y \in C^{\mathcal{I}} \mid \langle x, y\rangle \in S^{\mathcal{I}}\} \leq n\}$
where $\mathsf{card}\{S\}$ *denotes the cardinality of a set S.*

$*$ *$C \sqsubseteq D$ is called a general concept inclusion (GCI) where C, D are $\mathcal{SHIQ}_{+}$-concepts (possibly complex), and a finite set of GCIs is called a terminology $\mathcal{T}$. An interpretation $\mathcal{I}$ satisfies a GCI $C \sqsubseteq D$ if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ and $\mathcal{I}$ satisfies a terminology $\mathcal{T}$ if $\mathcal{I}$ satisfies each GCI in $\mathcal{T}$. Such an interpretation is called a* model *of $\mathcal{T}$, denoted by $\mathcal{I} \models \mathcal{T}$.*

$*$ *A concept C is called* satisfiable *w.r.t. a role hierarchy $\mathcal{R}$ and a terminology $\mathcal{T}$ iff there is some interpretation $\mathcal{I}$ such that $\mathcal{I} \models \mathcal{R}$, $\mathcal{I} \models \mathcal{T}$ and $C^{\mathcal{I}} \neq \emptyset$. Such an interpretation is called a* model *of C w.r.t. $\mathcal{R}$ and $\mathcal{T}$. A pair $(\mathcal{T}, \mathcal{R})$ is called a $\mathcal{SHIQ}_{+}$ knowledge base and said to be consistent if there is a model $\mathcal{I}$ of both $\mathcal{T}$ and $\mathcal{R}$, i.e., $\mathcal{I} \models \mathcal{T}$ and $\mathcal{I} \models \mathcal{R}$.*

$*$ *A concept D* subsumes *a concept C w.r.t. $\mathcal{R}$ and $\mathcal{T}$, denoted by $C \sqsubseteq D$, if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ holds in each model $\mathcal{I}$ of $(\mathcal{T}, \mathcal{R})$.*

Notice that we can reduce subsumption and consistency problems in $\mathcal{SHIQ}_{+}$ to concept satisfiability w.r.t. a knowledge base $(\mathcal{T}, \mathcal{R})$. Thanks to these reductions, it suffices to study the concept satisfiability problem in $\mathcal{SHIQ}_{+}$.

For the ease of construction, we assume all concepts to be in *negation normal form* (NNF) i.e. negation occurs only in front of concept names. Any $\mathcal{SHIQ}_{+}$-concept can be transformed to an equivalent one in NNF by using DeMorgan's laws and some equivalences as presented in [10]. For a concept C, we denote the nnf of C by $\mathsf{nnf}(C)$ and the nnf of $\neg C$ by $\dot{\neg} C$

Let D be an $\mathcal{SHIQ}_{+}$-concept in NNF. We define $\mathsf{sub}(D)$ to be the smallest set that contains all sub-concepts of D including D. For a knowledge base $(\mathcal{T}, \mathcal{R})$, $\mathbf{R}_{(\mathcal{T},\mathcal{R})}$ is used to denote the set of all role names occurring in $\mathcal{T}, \mathcal{R}$ with their transitive closure and inverses. We denote by $\mathbf{R}^{+}_{(\mathcal{T},\mathcal{R})}$ the set of transitive closure of roles occurring in

$\mathbf{R}_{(\mathcal{T},\mathcal{R})}$. Finally, we define sets $\mathsf{sub}(\mathcal{T},\mathcal{R})$ and $\widehat{\mathsf{sub}}(\mathcal{T},\mathcal{R})$ as follows:

$$\mathsf{sub}(\mathcal{T},\mathcal{R}) = \bigcup_{C \sqsubseteq D \in \mathcal{T}} \mathsf{sub}(\mathsf{nnf}(\neg C \sqcup D),\mathcal{R}) \textit{ where} \tag{1}$$

$$\begin{aligned}\mathsf{sub}(E,\mathcal{R}) = {} & \mathsf{sub}(E) \cup \{\dot\neg C \mid C \in \mathsf{sub}(E)\} \cup \\ & \{\forall S.C \mid (\forall R.C \in \mathsf{sub}(E), S \underline{\boxast} R) \textit{ or } (\dot\neg \forall R.C \in \mathsf{sub}(E), S \underline{\boxast} R) \\ & \qquad \textit{where } S \textit{ occurs in } \mathcal{T} \textit{ or } \mathcal{R}\} \cup \\ & \{\exists P.\beta \mid \beta \in \{C, \exists P^{\oplus}.C\}, \exists P^{\oplus}.C \in \mathsf{sub}(E)\}\end{aligned} \tag{2}$$

$$\Phi_\sigma = \bigsqcap_{C \in \sigma \,\cup\, \{\dot\neg D \mid D \in \mathsf{sub}(\mathcal{T},\mathcal{R}) \setminus \sigma\}} C \;\textit{ for each } \sigma \subseteq \mathsf{sub}(\mathcal{T},\mathcal{R}) \tag{3}$$

$$\Omega = \{\Phi_\sigma \mid \sigma \subseteq \mathsf{sub}(\mathcal{T},\mathcal{R})\} \tag{4}$$

$$\widehat{\mathsf{sub}}(\mathcal{T},\mathcal{R}) = \Omega \cup \{\alpha.\beta \mid \alpha \in \{\exists P.\exists P^{\oplus}, \exists P^{\oplus}, \exists P\}, P^{\oplus} \in \mathbf{R}^{+}_{(\mathcal{T},\mathcal{R})}, \beta \in \Omega\} \tag{5}$$

3 Tableaux for $\mathcal{SHIQ}_+$

Basically, a tableau structure is used to represent a model of a $\mathcal{SHIQ}_+$ knowledge base. Properties in such a tableau definition express semantic constraints resulting directly from the logic constructors in $\mathcal{SHIQ}_+$. Considering the tableau definition for $\mathcal{SHIQ}$ presented in [10], Definition 3 for $\mathcal{SHIQ}_+$ adopts two additional properties, namely P8 and P9. In particular, P8 imposes a global constraint on a set of individuals of a tableau. This causes the tableaux to lose *the local satisfiability property*. A tableau has the local satisfiability property if each property of the tableau is related to only one node and its neighbors. This means that, for a graph with a labelling function, checking each node of the graph and its neighbors for each property is sufficient to prove whether this graph is a tableau. The tableau definition for $\mathcal{SHIQ}$ in [10] has the local satisfiability property although $\mathcal{SHIQ}$ includes transitive roles. The propagation of value restrictions on transitive roles by $\forall^{+}$-rule (i.e. the rule for $\forall R.C$ if R is transitive or includes a transitive role) and the absence of number restrictions on transitive roles help to avoid global properties that impose a constraint on an arbitrary set of individuals in a tableau.

Definition 3. *Let $(\mathcal{T},\mathcal{R})$ be a $\mathcal{SHIQ}_+$ knowledge base. A tableau T for a concept D w.r.t $(\mathcal{T},\mathcal{R})$ is defined to be a triplet $(\mathbf{S},\mathcal{L},\mathcal{E})$ such that $\mathbf{S}$ is a set of individuals, $\mathcal{L}$: $\mathbf{S} \rightarrow 2^{\mathsf{sub}(\mathcal{T},\mathcal{R}) \cup \widehat{\mathsf{sub}}(\mathcal{T},\mathcal{R})}$ and $\mathcal{E}$: $\mathbf{R}_{(\mathcal{T},\mathcal{R})} \rightarrow 2^{\mathbf{S}\times\mathbf{S}}$, and there is some individual $s \in \mathbf{S}$ such that $D \in \mathcal{L}(s)$. For all $s \in \mathbf{S}$, $C, C_1, C_2 \in \mathsf{sub}(\mathcal{T},\mathcal{R}) \cup \widehat{\mathsf{sub}}(\mathcal{T},\mathcal{R})$, $R, S \in \mathbf{R}_{(\mathcal{T},\mathcal{R})}$ and $Q^{\oplus} \in \mathbf{R}^{+}_{(\mathcal{T},\mathcal{R})}$, T satisfies the following properties:*

P1 *If $C_1 \sqsubseteq C_2 \in \mathcal{T}$ then $\mathsf{nnf}(\neg C_1 \sqcup C_2) \in \mathcal{L}(s)$,*
P2 *If $C \in \mathcal{L}(s)$ then $\dot\neg C \notin \mathcal{L}(s)$,*
P3 *If $C_1 \sqcap C_2 \in \mathcal{L}(s)$ then $C_1 \in \mathcal{L}(s)$ and $C_2 \in \mathcal{L}(s)$,*
P4 *If $C_1 \sqcup C_2 \in \mathcal{L}(s)$ then $C_1 \in \mathcal{L}(s)$ or $C_2 \in \mathcal{L}(s)$,*
P5 *If $\forall S.C \in \mathcal{L}(s)$ and $\langle s,t\rangle \in \mathcal{E}(S)$ then $C \in \mathcal{L}(t)$,*
P6 *If $\forall S.C \in \mathcal{L}(s)$, $Q^{\oplus} \underline{\boxast} S$ and $\langle s,t\rangle \in \mathcal{E}(Q)$ then $\forall Q^{\oplus}.C \in \mathcal{L}(t)$,*

P7 *If* $\exists P.C \in \mathcal{L}(s)$ *with* $P \in \mathbf{R}_{(\mathcal{T},\mathcal{R})} \setminus \mathbf{R}^+_{(\mathcal{T},\mathcal{R})}$ *then there is some* $t \in \mathbf{S}$ *such that* $\langle s,t\rangle \in \mathcal{E}(P)$ *and* $C \in \mathcal{L}(t)$,

P8 *If* $\exists Q^{\oplus}.C \in \mathcal{L}(s)$ *then* $(\exists Q.C \sqcup \exists Q.\exists Q^{\oplus}.C) \in \mathcal{L}(s)$, *and there are* $s_1, \cdots, s_n \in \mathbf{S}$ *such that* $\exists Q.C \in \mathcal{L}(s_0) \cup \mathcal{L}(s_{n-1})$ *and* $\langle s_i, s_{i+1}\rangle \in \mathcal{E}(Q)$ *with* $0 \le i < n$, $s_0 = s$ *and* $\exists Q^{\oplus}.C \in \mathcal{L}(s_j)$ *for all* $0 \le j < n$.

P9 *If* $\langle s,t\rangle \in \mathcal{E}(Q^{\oplus})$ *then* $\exists Q^{\oplus}.\Phi_\sigma \in \mathcal{L}(s)$ *with* $\sigma = \mathcal{L}(t) \cap \mathsf{sub}(\mathcal{T},\mathcal{R})$ *and*

$$\Phi_\sigma = \bigsqcap_{C \in \sigma \,\cup\, \{\dot{\neg} D | D \in \mathsf{sub}(\mathcal{T},\mathcal{R}) \setminus \sigma\}} C,$$

P10 $\langle s,t\rangle \in \mathcal{E}(R)$ *iff* $\langle t,s\rangle \in \mathcal{E}(R^{\ominus})$,

P11 *If* $\langle s,t\rangle \in \mathcal{E}(R)$ *and* $R \underline{\boxast} S$ *then* $\langle s,t\rangle \in \mathcal{E}(S)$,

P12 *If* $(\le nS.C) \in \mathcal{L}(s)$ *then* $\mathsf{card}\{S^T(s,C)\} \le n$ *where* $S^T(s,C) := \{t \in \mathbf{S} | \langle s,t\rangle \in \mathcal{E}(S) \wedge C \in \mathcal{L}(t)\}$,

P13 *If* $(\ge nS.C) \in \mathcal{L}(s)$ *then* $\mathsf{card}\{S^T(s,C)\} \ge n$,

P14 *If* $(\le nS.C) \in \mathcal{L}(s)$ *and* $\langle s,t\rangle \in \mathcal{E}(S)$ *then* $C \in \mathcal{L}(t)$ *or* $\dot{\neg} C \in \mathcal{L}(t)$.

P8 in Definition 3 expresses not only the semantic constraint imposed by the transitive closure of roles occurring in concepts such as $\exists Q^{\oplus}.C$ (i.e. a path including nodes are connected by edges containing Q and the label of the last node contains C) but also the non-determinism of transitive closure of roles (i.e. the term $\exists Q.C$ may be chosen at any node of such a path to satisfy $\exists Q^{\oplus}.C$). Additionally, P8 and P9 in Definition 3 enable to satisfy each transitive closure $Q^{\oplus}$ occurring in the label of an edge $\langle s,t\rangle$ with simple role Q. In fact, P9 makes Φ_σ belong to the label of a node t' and s connected to t' by edges containing Q due to P8. The definition of Φ_σ allows t' to be combined with t without causing contradiction. Moreover, this combination does not violate number restrictions since Q is simple. For this reason, the new definition for simple roles presented in Definition 1 is crucial to decidability of $\mathcal{SHIQ}_+$.

In addition, P8 and P9 defined in this way do not require explicitly cycles to satisfy role inclusion axioms such as $R \sqsubseteq Q^{\oplus}$. This makes it possible to design of tableaux-based algorithm for $\mathcal{SHIQ}_+$ that aims to build tree-like structure i.e. no cycle is explicitly required to be embedded within this structure. The following lemma affirms that a tableau represents exactly a model for the concept.

Lemma 1. *Let* $(\mathcal{T},\mathcal{R})$ *be a* $\mathcal{SHIQ}_+$ *knowledge base. Let* D *be a* $\mathcal{SHIQ}_+$ *concept.* D *is satisfiable w.r.t.* $(\mathcal{T},\mathcal{R})$ *iff there is a tableau for* D *w.r.t.* $(\mathcal{T},\mathcal{R})$.

For a proof of Lemma 1, we refer the reader to [13].

4 A Tableaux-Based Decision Procedure for $\mathcal{SHIQ}_+$

As mentioned, a tableau for a concept represents a model that is possibly infinite. However, the goal of a tableaux-based algorithm is to find a finite structure that must imply a tableau. Conversely, the existence of a tableau can guide us to build such a structure. We introduce in Definition 4 such a finite structure, namely, completion tree.

Definition 4. *Let* $(\mathcal{T},\mathcal{R})$ *be a* $\mathcal{SHIQ}_+$ *knowledge base. Let* D *be a* $\mathcal{SHIQ}_+$ *concept. A* completion tree *for* D *and* $(\mathcal{T},\mathcal{R})$ *is a tree* $\mathbf{T} = (V, E, \mathcal{L}, x_{\mathbf{T}}, \neq)$ *where*

$*$ *V is a set of nodes containing a root node $x_{\mathbf{T}} \in V$. Each node $x \in V$ is labelled with a function $\mathcal{L}$ such that $\mathcal{L}(x) \subseteq \mathsf{sub}(\mathcal{T},\mathcal{R}) \cup \widehat{\mathsf{sub}}(\mathcal{T},\mathcal{R})$. In addition, $\neq$ is a symmetric binary relation over V.*

$*$ *E is a set of edges. Each edge $\langle x,y\rangle \in E$ is labelled with a function $\mathcal{L}$ such that $\mathcal{L}(\langle x,y\rangle) \subseteq \mathbf{R}_{(\mathcal{T},\mathcal{R})}$.*

$*$ *If $\langle x,y\rangle \in E$ then y is called a successor of x, denoted by $y \in \mathsf{succ}^1(x)$, or x is called the predecessor of y, denoted by $x = \mathsf{pred}^1(y)$. In this case, we say that x is a neighbor of y or y is a neighbor of x. If $z \in \mathsf{succ}^n(x)$ (resp. $z = \mathsf{pred}^n(x)$) and y is a successor of z (resp. y is the predecessor of z) then $y \in \mathsf{succ}^{(n+1)}(x)$ (resp. $y = \mathsf{pred}^{(n+1)}(x)$) for all $n \geq 0$ where $\mathsf{succ}^0(x) = \{x\}$ and $\mathsf{pred}^0(x) = x$.*

$*$ *A node y is called a R-successor of x, denoted by $y \in \mathsf{succ}^1_R(x)$ (resp. y is called the R-predecessor of x, denoted by $y = \mathsf{pred}^1_R(x)$) if there is some role R' such that $R' \in \mathcal{L}(\langle x,y\rangle)$ (resp. $R' \in \mathcal{L}(\langle y,x\rangle)$) and $R' \underline{\boxed{*}} R$. A node y is called a R-neighbor of x if y is either a R-successor or R-predecessor of x. If z is a R-successor of y (resp. z is the R-predecessor of y) and $y \in \mathsf{succ}^n_R(x)$ (resp. $y = \mathsf{pred}^n_R(x)$) then $z \in \mathsf{succ}^{(n+1)}_R(x)$ (resp. $z = \mathsf{pred}^{(n+1)}_R(x)$) for $n \geq 0$ with $\mathsf{succ}^0_R(x) = \{x\}$ and $x = \mathsf{pred}^0_R(x)$.*

$*$ *For a node x and a role S, we define the set $S^{\mathbf{T}}(x,C)$ of x's S-neighbors as follows:*

$$S^{\mathbf{T}}(x,C) = \{y \in V \mid y \text{ is a } S\text{-neighbor of } x \text{ and } C \in \mathcal{L}(x)\}$$

$*$ *A node x is called* blocked *by y, denoted by $y = \mathsf{b}(x)$, if there are numbers $n, m > 0$ and nodes x', y, y' such that*

1. *$x_{\mathbf{T}} = \mathsf{pred}^n(y)$, $y = \mathsf{pred}^m(x)$, and*
2. *$x' = \mathsf{pred}^1(x)$, $y' = \mathsf{pred}^1(y)$, and*
3. *$\mathcal{L}(x) = \mathcal{L}(y)$, $\mathcal{L}(x') = \mathcal{L}(y')$, and*
4. *$\mathcal{L}(\langle x',x\rangle) = \mathcal{L}(\langle y',y\rangle)$, and*
5. *if there are z, z' such that $z' = \mathsf{pred}^1(z)$, $\mathsf{pred}^i(z') = x_{\mathbf{T}}$, $\mathcal{L}(z) = \mathcal{L}(y)$, $\mathcal{L}(z') = \mathcal{L}(y')$ and $\mathcal{L}(\langle z',z\rangle) = \mathcal{L}(\langle y',y\rangle)$ then $n \leq i$.*

$*$ *We define an extended function $\widehat{\mathsf{succ}}$ from succ over $\mathbf{T}$ as follows:*

- *if x has a successor y (resp. x has a R-successor y) that is not blocked then $y \in \widehat{\mathsf{succ}}^1(x)$ (resp. $y \in \widehat{\mathsf{succ}}^1_R(x)$),*
- *if x has a successor z (resp. x has a R-successor z) that is blocked by $\mathsf{b}(z)$ then $\mathsf{b}(z) \in \widehat{\mathsf{succ}}^1(x)$ (resp. $\mathsf{b}(z) \in \widehat{\mathsf{succ}}^1_R(x)$).*
- *if $y \in \widehat{\mathsf{succ}}^n_R(x)$ and $z \in \widehat{\mathsf{succ}}^1_R(y)$ then $z \in \widehat{\mathsf{succ}}^{(n+1)}_R(x)$ for $n \geq 0$.*

$*$ *A node z is called a $\exists R^{\oplus}.C$-reachable of x with $\exists R^{\oplus}.C \in \mathcal{L}(x)$ if there are $x_1, \cdots, x_{k+n} \in V$ with $x_{k+n} = z, x_0 = x$ and $k + n \geq 0$ such that $x_i = \mathsf{pred}^i_R(x_0)$, $\exists R^{\oplus}.C \in \mathcal{L}(x_i)$ with $i \in \{0, \cdots, k\}$, and $x_{j+k} \in \widehat{\mathsf{succ}}^j_R(x_k)$, $\exists R^{\oplus}.C \in \mathcal{L}(x_{j+k})$, $\exists R.C \in \mathcal{L}(x_{(k+n)})$ with $j \in \{0, \cdots, n\}$.*

$*$ **Clashes** : $\mathbf{T}$ *is said to contain a* clash *if one of the following conditions holds:*

1. *There is some node $x \in V$ such that $\{A, \dot{\neg}A\} \subseteq \mathcal{L}(x)$ for some concept name $A \in \mathbf{C}$,*

2. *There is some node $x \in V$ with $(\leq nS.C) \in \mathcal{L}(x)$ and there are $(n+1)$ S-neighbors $y_1, \cdots, y_{n+1}$ of x such that $y_i \dot{\neq} y_j$ and $C \in \mathcal{L}(x_i)$ for all $1 \leq i < j \leq (n+1)$,*
3. *There is some node $x \in V$ with $\exists R^{\oplus}.C \in \mathcal{L}(x)$ such that there does not exist any $\exists R^{\oplus}.C$-reachable node y of x,*

Algorithm 2 builds a completion tree for a $\mathcal{SHIQ}_+$ concept by applying the expansion rules in Figure 2 and 3. The expansion rules in Figure 2 were given in [10]. We introduce two new expansion rules that correspond to P8 and P9 in Definition 3.

In comparison with $\mathcal{SHIQ}$, there is a new source of non-determinisms that could augment the complexity of an algorithm for satisfiability of concepts in $\mathcal{SHIQ}_+$. This source comes from the presence of transitive closure of role in concepts. This means that for each occurrence of a term such as $\exists Q^{\oplus}.C$ in the label of a node of a completion tree we have to check the existence of a sequence of edges such that the label of each edge contains Q and the label of the last node contains C. The process for checking the existence of paths whose length is arbitrary must be translated into a process that works for a finite structure. To do this, we reuse the blocking condition introduced in [10] and introduce a function $\widehat{\mathsf{succ}}(x)$ that returns the set of x's successors in a completion tree. An infinite path over a completion tree can be defined thanks to this function. The $\exists_+$-rule in Figure 3 generates all possible paths. The clash-freeness of the third kind in Definition 4 ensures that a "good" path has to be picked from this set of all possible paths.

The function $\mathsf{checkReachability}_C^Q(x, d, \mathcal{B})$ depicted in Algorithm 1 represents an algorithm for checking the clash-freeness of the third kind for a completion tree. It returns true iff there exists a $\exists Q^{\oplus}.C$-reachable node of x. In this function, the parameter x represents a node of the tree to be checked i.e. there is a term such as $\exists Q^{\oplus}.C \in \mathcal{L}(x)$. The parameter d indicates the direction to search from x. Depending on $d = 1$ or $d = 0$, the algorithm goes up to ancestors of x or goes down to descendants of x respectively. When the algorithm goes down, it never goes up again. The subset $\mathcal{B} \subseteq V$ represents the set of all blocked nodes among the nodes that the algorithm has visited. The function $\mathsf{checkReachability}_C^Q(x, 1, \emptyset)$ would be called for each non-blocked node x of a completion tree and for each term of the form $\exists Q^{\oplus}.C \in \mathcal{L}(x)$.

Lemma 2 (Termination). *Let $(\mathcal{T}, \mathcal{R})$ be a $\mathcal{SHIQ}_+$ knowledge base. Let D be a $\mathcal{SHIQ}_+$-concept w.r.t. $(\mathcal{T}, \mathcal{R})$. Algorithm 2 terminates.*

Proof. The termination of Algorithm 2 is a consequence of the following claims:

1. Applications of rules in Figure 2 and 3 do not remove concepts from the label of nodes. Moreover, applications of rules in Figure 2 and 3 do not remove roles from the label of edges except that they may set the label of edges to an empty set. However, when the label of an edge becomes empty it remains to be empty forever. Therefore, we can compute a upper bound of the completion tree's height from the blocking condition. This upper bound equals $K = 2^{2m+k}$ where $m = \mathsf{card}\{\mathsf{sub}(\mathcal{T}, \mathcal{R}) \cup \widehat{\mathsf{sub}}(\mathcal{T}, \mathcal{R})\}$ and k is the number of roles occurring in $\mathcal{T}$ and $\mathcal{R}$ plus their inverse and transitive closure. Moreover, the number of neighbors of any node is bounded by $M = \sum m_i$ where m_i occurs in a number restriction term $(\geq m_i R.C)$ that appears in $\mathcal{T}$.

```
$\sqsubseteq$-rule: if   $C \sqsubseteq D \in \mathcal{T}$ and $\mathsf{nnf}(\neg C \sqcup D) \notin \mathcal{L}(x)$
          then $\mathcal{L}(x) \longleftarrow \mathcal{L}(x) \cup \{\mathsf{nnf}(\neg C \sqcup D)\}$
$\sqcap$-rule: if   $C_1 \sqcap C_2 \in \mathcal{L}(x)$ and $\{C_1, C_2\} \not\subseteq \mathcal{L}(x)$
          then $\mathcal{L}(x) \longleftarrow \mathcal{L}(x) \cup \{C_1, C_2\}$
$\sqcup$-rule: if   $C_1 \sqcup C_2 \in (x)$ and $\{C_1, C_2\} \cap \mathcal{L}(x) = \emptyset$
          then $\mathcal{L}(x) \longleftarrow \mathcal{L}(x) \cup \{C\}$ for some $C \in \{C_1, C_2\}$
$\exists$-rule: if 1. $\exists S.C \in \mathcal{L}(x)$, $x$ is not blocked, and
             2. $x$ has no $S$-neighbour $y$ with $C \in \mathcal{L}(y)$
          then create a new node $y$ with $\mathcal{L}(\langle x, y\rangle)$=$\{S\}$ and $\mathcal{L}(y)$=$\{C\}$
$\forall$-rule: if 1. $\forall S.C \in \mathcal{L}(x)$, and
             2. there is a $S$-neighbour $y$ of $x$ such that $C \notin \mathcal{L}(y)$
          then $\mathcal{L}(y) \longleftarrow \mathcal{L}(y) \cup \{C\}$
$\forall_+$-rule: if 1. $\forall S.C \in \mathcal{L}(x)$, and
             2. there is some $Q$ with $Q^{\oplus} \underline{\boxast} S$, and
             3. there is an $Q$-neighbour $y$ of $x$ such that $\forall Q^{\oplus}.C \notin \mathcal{L}(y)$
          then $\mathcal{L}(y) \longleftarrow \mathcal{L}(y) \cup \{\forall Q^{\oplus}.C\}$
$ch$-rule: if 1. $(\leq n\, S.C) \in \mathcal{L}(x)$, and
             2. there is an $S$-neighbour $y$ of $x$ with $\{C, \dot{\neg} C\} \cap \mathcal{L}(y) = \emptyset$
          then $\mathcal{L}(y) \longleftarrow \mathcal{L}(y) \cup \{E\}$ for some $E \in \{C, \dot{\neg} C\}$
$\geq$-rule: if 1. $(\geq n\, S.C) \in \mathcal{L}(x)$ and $x$ is not blocked, and
             2. there are no $n$ $S$-neighbors $y_1, ..., y_n$ such that $C \in \mathcal{L}(y_i)$, and $y_i \neq y_j$ for
                $1 \leq i < j \leq n$,
          then create $n$ new nodes $y_1, ..., y_n$ with $\mathcal{L}(\langle x, y_i\rangle)$=$\{S\}$,
                $\mathcal{L}(y_i)$=$\{C\}$, and $y_i \neq y_j$ for $1 \leq i < j \leq n$.
$\leq$-rule: if 1. $(\leq n\, S.C) \in \mathcal{L}(x)$, and
             2. $\mathsf{card}\{S^{\mathbf{T}}(x, C)\} > n$ and there are two $S$-neighbors $y, z$ of $x$ with
                $C \in \mathcal{L}(y) \cap \mathcal{L}(z)$, $y$ is not an ancestor of $z$, and not $y \neq z$
          then 1. $\mathcal{L}(z) \longleftarrow \mathcal{L}(z) \cup \mathcal{L}(y)$ and $\mathcal{L}(\langle x, y\rangle) \longleftarrow \emptyset$
                2. If $z$ is an ancestor of $x$
                   then $\mathcal{L}(\langle z, x\rangle) \longleftarrow \mathcal{L}(\langle z, x\rangle) \cup \{R^{\ominus} \mid R \in \mathcal{L}(\langle x, y\rangle)\}$
                   else $\mathcal{L}(\langle x, z\rangle) \longleftarrow \mathcal{L}(\langle x, z\rangle) \cup \mathcal{L}(\langle x, y\rangle)$
                4. Add $u \neq z$ for all $u$ such that $u \neq y$
```

Fig. 2. Expansion rules for $\mathcal{SHIQ}$ presented in [10]

```
$\exists_+$-rule: if   $\exists S^{\oplus}.C \in \mathcal{L}(x)$ and $(\exists S.C \sqcup \exists S.\exists S^{\oplus}.C) \notin \mathcal{L}(x)$
          then $\mathcal{L}(x) \longleftarrow \mathcal{L}(x) \cup \{\exists S.C \sqcup \exists S.\exists S^{\oplus}.C\}$
$\oplus$-rule: if   $x$ has a $P^{\oplus}$-neighbor $y$ and $\exists P^{\oplus}.\Phi_\sigma \notin \mathcal{L}(x)$ with $\sigma = \mathcal{L}(y) \cap \mathsf{sub}(\mathcal{T}, \mathcal{R})$
          then $\mathcal{L}(x) = \mathcal{L}(x) \cup \{\exists P^{\oplus}.\Phi_\sigma\}$
```

Fig. 3. New expansion rules for $\mathcal{SHIQ}_+$

```
checkReachability$^Q_C(x, d, \mathcal{B})$
if $\exists Q.C \in \mathcal{L}(x)$ then
    return true;
if $d = 1$ then
    if there is $\mathsf{pred}^1_Q(x)$ with $\exists Q^{\oplus}.C \in \mathcal{L}(\mathsf{pred}^1_Q(x))$ then
        checkReachability$^Q_C(\mathsf{pred}^1_Q(x), 1, \mathcal{B})$ ;
foreach $x' \in \mathsf{succ}^1_Q(x)$ such that $\exists Q^{\oplus}.C \in \mathcal{L}(x')$ do
    if $\exists Q.C \in \mathcal{L}(x')$ then
        return true;
    if $x'$ is not blocked then
        checkReachability$^Q_C(x', 0, \mathcal{B})$ ;
    else
        if $x' \notin \mathcal{B}$ then
            $\mathcal{B} = \mathcal{B} \cup \{x'\}$;
            checkReachability$^Q_C(\mathsf{b}(x'), 0, \mathcal{B})$ ;
return false;
```

Algorithm 1. checkReachability$^Q_C(x, d, \mathcal{B})$ for checking the existence of a $\exists Q^{\oplus}.C$-reachable node of $x \in V$ where $d \in \{1, 0\}$, $\mathcal{B} \subseteq V$, $\exists Q^{\oplus}.C \in \mathcal{L}(x)$ and $\mathbf{T} = (V, E, \mathcal{L}, x_{\mathbf{T}}, \dot{\neq})$ is a completion tree. As shown in Lemma 2, the complexity of Algorithm 1 is bounded by an exponential function in size of a completion tree. This implies that the complexity of the tableaux algorithm for $\mathcal{SHIQ}_+$ (Algorithm 2) is bounded by a double exponential function in size of inputs.

```
   Input  : A $\mathcal{SHIQ}_+$ knowledge base $(\mathcal{T}, \mathcal{R})$ and a $\mathcal{SHIQ}_+$-concept $D$
   Output: Is $D$ satisfiable w.r.t. $(\mathcal{T}, \mathcal{R})$ ?
1  Let $\mathbf{T} = (V, E, \mathcal{L}, x_{\mathbf{T}}, \dot{\neq})$ be an initial tree such that $V = \{x_{\mathbf{T}}\}$, $\mathcal{L}(x_{\mathbf{T}}) = \{D\}$, and
   there is no $x, y \in V$ such that $x \dot{\neq} y$;
2  while there is a non-empty set S of expansion rules in Figure 2 and 3 such that each r $\in$ S
   can be applied to a node $x \in V$ do
3      Apply r ;
4  if there is a clash-free tree $\mathbf{T}'$ which is built by Line 2 to 2 then
5      YES ;
6  else
7      NO ;
```

Algorithm 2. Algorithm for building a completion tree in $\mathcal{SHIQ}_+$

2. Algorithm 1 checks the clash-freeness of the third kind for each $x \in V$ with $\exists Q^{\oplus}.C \in \mathcal{L}(x)$. To do this, it starts from x and go up to an ancestor x' of x, and go down to a descendant of x' through the function $\mathsf{succ}(x')$. The length of such a path is bounded by $K \times L$ where K is given above and L is the number of blocked nodes of the completion tree. Algorithm 1 may consider all paths which go though all possible blocked nodes. The cardinality of this set is bounded by the number of all permutations of the blocked nodes. Therefore, the complexity of Algorithm 1 is bounded by $(K \times L) \times L!$. Algorithm 1 would be called for each occurrence of each term such as $\exists Q^{\oplus}.C$ that occurs in each node $v \in V$.

Lemma 3 (Soundness). *Let $(\mathcal{T}, \mathcal{R})$ be a $\mathcal{SHIQ}_+$ knowledge base. Let D be a $\mathcal{SHIQ}_+$-concept w.r.t. $(\mathcal{T}, \mathcal{R})$. If Algorithm 2 can build a clash-free completion tree for D w.r.t. $(\mathcal{T}, \mathcal{R})$ then there is a tableau for D w.r.t. $(\mathcal{T}, \mathcal{R})$.*

Proof sketch. Assume that $\mathbf{T} = (V, E, \mathcal{L}, x_{\mathbf{T}}, \neq)$ is a clash-free completion tree for D w.r.t. $(\mathcal{T}, \mathcal{R})$. First, we build an extended tree $\widehat{\mathbf{T}} = (\widehat{V}, \widehat{E}, \mathcal{L}, x_{\widehat{\mathbf{T}}}, \neq)$ from $\mathbf{T}$ with help of functions $\widehat{\mathsf{succ}}$ and $\mathsf{b}(x)$ as follows:

We define $x_{\widehat{\mathbf{T}}} = x_{\mathbf{T}}$. If $x \in \widehat{V}$ and $x' \in \widehat{\mathsf{succ}}(x)$ then we add to $\widehat{V}$ a successor x' of x. In particular, if z, z' are two distinct successors of x such that $\mathsf{b}(z) = \mathsf{b}(z')$ then there are two distinct nodes that are added to $\widehat{V}$. We define a tableau $T = (\mathbf{S}, \mathcal{L}', \mathcal{E})$ for D as follows:

- We define $\mathbf{S} = \widehat{V} = \bigcup\limits_{n \geq 0} \widehat{\mathsf{succ}}^{n}(x_{\mathbf{T}})$,
- For each $s \in \widehat{V}$ there is a unique $x_s \in V$ such that $x_s \in \mathsf{succ}^{k}(x_{\mathbf{T}})$ and $s \in \widehat{\mathsf{succ}}^{l}(x_s)$ with $n = k + l$. We define $\mathcal{L}'(s) = \mathcal{L}(x_s)$.
- $\mathcal{E}(R) = \mathcal{E}_1(R) \cup \mathcal{E}_2(R)$ where
  $\mathcal{E}_1(R) = \{\langle s, t\rangle \in \mathbf{S}^2 \mid R \in \mathcal{L}(\langle x_s, x_t\rangle) \vee R^{\ominus} \in \mathcal{L}(\langle x_t, x_s\rangle)\}$, and
  $\mathcal{E}_2(R) = \{\langle s, t\rangle \in \mathbf{S}^2 \mid (R \in \mathcal{L}(\langle x_s, z\rangle) \wedge (\mathsf{b}(z) = x_t)\ ) \vee (R^{\ominus} \in \mathcal{L}(\langle x_t, z'\rangle) \wedge (\mathsf{b}(z') = x_s)\ )\}$

We now show that T satisfies P8 in Definition 3, which is the most problematical property. For the other properties, we refer the reader to [13].

Assume that $s \in \mathbf{S}$ with $\exists Q^{\oplus}.C \in \mathcal{L}'(s)$. Since $\mathbf{T}$ is clash-free (third kind), x_s has a $\exists Q^{\oplus}.C$-reachable x_n i.e. there are $x_1, \cdots, x_n$ such that x_{i+1} is a Q-neighbor of x_i or x_{i+1} blocks a Q-successor of x_i with $x_s = x_0$ and $\exists Q.C \in \mathcal{L}(x_n)$, $\exists Q^{\oplus}.C \in \mathcal{L}(x_i)$ for all $i \in \{0, \cdots, n-1\}$.

Assume that $\exists Q.C \in \mathcal{L}'(s)$. This implies that x_s has a Q-neighbor y such that $C \in \mathcal{L}(y)$ due to the non-applicable of $\exists$-rule. By the definition of T, there is some $t \in \mathbf{S}$ with $t \in \widehat{\mathsf{succ}}^{1}(s)$ or $s \in \widehat{\mathsf{succ}}^{1}(t)$ such that $\langle s, t\rangle \in \mathcal{E}(Q)$. Thus, P8 holds.

Assume that $\exists Q.C \notin \mathcal{L}'(s)$. According to the definition of $\exists Q^{\oplus}.C$-reachable nodes, there is some $0 \leq k < n$ such that x_k is an ancestor of x_0 and x_{k+1} is a (extended) successor of x_k. If $k = 0$ then there are $s_1, \cdots, s_n$ with $x_{s_i} = x_i$, $s_0 = s$ and $\langle s_i, s_{i+1}\rangle \in \mathcal{E}(Q)$, $\exists Q.C \in \mathcal{L}'(s_n)$, $\exists Q^{\oplus}.C \in \mathcal{L}'(s_i)$ for all $i \in \{0, \cdots, n\}$. Thus, P8 holds. Assume that $k > 0$. We define a function $\widehat{\mathsf{pred}}^{j}(t)$ as follows: $\widehat{\mathsf{pred}}^{j}(t) = x_{\mathbf{T}}$ iff $t \in \widehat{\mathsf{succ}}^{j}(x_{\mathbf{T}})$ for all $t \in \mathbf{S}$. This implies that for each $t \in \mathbf{S}$ there is a unique

j such that $\widehat{\mathsf{pred}}^{j}(t) = x_{\mathbf{T}}$. Let $x_{\mathbf{T}} = \widehat{\mathsf{pred}}^{l}(s)$, $x_{\mathbf{T}} = \widehat{\mathsf{pred}}^{m}(x_0) = \mathsf{pred}^{m}(x_0)$ and $x_{\mathbf{T}} = \widehat{\mathsf{pred}}^{p}(x_k) = \mathsf{pred}^{p}(x_k)$. We consider the following cases :

Assume $m = l$. By the definition of T there are $s_0, \cdots, s_n \in \mathbf{S}$ such that $x_{s_i} = x_i$ and $\langle s_i, s_{i+1}\rangle \in \mathcal{E}(Q)$, $\exists Q^{\oplus}.C \in \mathcal{L}'(s_i)$ for all $i \in \{0, \cdots, n-1\}$ with $s_0 = s$ and $\exists Q.C \in \mathcal{L}'(s_n)$. Thus, P8 holds.

Assume $m < l$. Let $0 \leq K \leq l$ be the least number such that $x_{\widehat{\mathsf{pred}}^{K}(s)}$ has a $\exists Q^{\oplus}.C$-reachable y with $y \in \widehat{\mathsf{succ}}^{K'}(x_{\widehat{\mathsf{pred}}^{K}(s)})$. We can pick $K = l - p$ with $x_{\mathbf{T}} = \widehat{\mathsf{pred}}^{p}(x_k)$ if there is no such K with $K < l - p$. If $K = 0$ then $k = 0$, which was considered. For $K > 0$, we show that $\langle \widehat{\mathsf{pred}}^{j+1}(s), \widehat{\mathsf{pred}}^{j}(s)\rangle \in \mathcal{E}(Q)$ and $\exists Q^{\oplus}.C \in \mathcal{L}'(\widehat{\mathsf{pred}}^{j+1}(s))$ for all $j \in \{0, \cdots, K-1\}$ (***).

For $j = 0$, we have $\langle s, \widehat{\mathsf{pred}}^{1}(s)\rangle \in \mathcal{E}(Q)$ and $\exists Q^{\oplus}.C \in \mathcal{L}'(\widehat{\mathsf{pred}}^{1}(s))$, since $\langle s, \widehat{\mathsf{pred}}^{1}(s)\rangle \notin \mathcal{E}(Q)$ or $\exists Q^{\oplus}.C \notin \mathcal{L}'(\widehat{\mathsf{pred}}^{1}(s))$ implies $K = 0$.

Assume that $\exists Q^{\oplus}.C \in \mathcal{L}'(\widehat{\mathsf{pred}}^{j}(s))$ with $j < K$. Due to the clash-freeness (third kind) of $\mathbf{T}$, $x_{\widehat{\mathsf{pred}}^{j}(s)}$ has a $\exists Q^{\oplus}.C$-reachable node w i.e. there are nodes $w_1, \cdots, w_{n'}$ and some $k' \geq 0$ such that $w_{k'}$ is an ancestor of $x_{\widehat{\mathsf{pred}}^{j}(s)}$, $w_{k'+1}$ is a (extended) successor of $w_{k'}$, w_i is a Q-neighbor of w_{i-1} and $\exists Q^{\oplus}.C \in \mathcal{L}(w_i)$, $\exists Q.C \in \mathcal{L}(w_{n'})$ for all $i \in \{1, \cdots, n'\}$ with $w_0 = x_{\widehat{\mathsf{pred}}^{j}(s)}$. Due to $j < K$ and $\mathcal{L}'(\widehat{\mathsf{pred}}^{j+1}(s)) = \mathcal{L}(x_{\widehat{\mathsf{pred}}^{j+1}(s)})$, we have $k' > 0$, $\langle \widehat{\mathsf{pred}}^{j+1}(s), \widehat{\mathsf{pred}}^{j}(s)\rangle \in \mathcal{E}(Q)$ and
$\exists Q^{\oplus}.C \in \mathcal{L}'(\widehat{\mathsf{pred}}^{j+1}(s))$. Thus, (***) holds.

From (***), it follows that there are $s_i = \widehat{\mathsf{pred}}^{i}(s)$ for all $i \in \{0, \cdots, K\}$ and $s_{K+j} = \widehat{\mathsf{succ}}^{j}(\widehat{\mathsf{pred}}^{K}(s))$ for all $j \in \{1, \cdots, K'\}$ such that $\langle s_h, s_{h+1}\rangle \in \mathcal{E}(Q)$ and $\exists Q^{\oplus}.C \in \mathcal{L}'(s_h)$, $\exists Q.C \in \mathcal{L}'(s_{K+K'})$ for all $h \in \{0, \cdots, K+K'\}$ with $s_0 = s$. Thus, P8 holds. $\triangleleft$

Lemma 4 (Completeness). *Let $(\mathcal{T}, \mathcal{R})$ be a $\mathcal{SHIQ}_+$ knowledge base. Let D be a $\mathcal{SHIQ}_+$-concept w.r.t. $(\mathcal{T}, \mathcal{R})$. If there is a tableau for D w.r.t. $(\mathcal{T}, \mathcal{R})$ then Algorithm 2 can build a clash-free completion tree for D w.r.t. $(\mathcal{T}, \mathcal{R})$.*

Proof sketch. Let $T = (\mathbf{S}, \mathcal{L}', \mathcal{E})$ be a tableau for D w.r.t. $(\mathcal{T}, \mathcal{R})$. We show that there exists a sequence of expansion rule applications such that it generates a clash-free completion tree $\mathbf{T} = (V, E, \mathcal{L}, x_{\mathbf{T}}, \neq)$ (**). We define a function π from V to $\mathbf{S}$ progressively over the construction of $\mathbf{T}$ such that it satisfies the following conditions, denoted by (*):

1. $\mathcal{L}(x) \subseteq \mathcal{L}'(\pi(x))$ for $x \in V$,
2. if y is a S-neighbor of x in $\mathbf{T}$ then $\langle \pi(x), \pi(y)\rangle \in \mathcal{E}(S)$,
3. $x \neq y$ implies $\pi(x) \neq \pi(y)$,
4. if $\exists Q^{\oplus}.C \in \mathcal{L}(x)$ and $\exists Q.C \in \mathcal{L}'(\pi(x))$ then $\exists Q.C \in \mathcal{L}(x)$ for $x \in V$,

To prove (**), we have to show that (i) we can apply expansion rules such that the conditions in (*) are preserved, and (ii) if the conditions (*) are satisfied when constructing a completion tree by expansion rules then the obtained completion tree is clash-free. Since T is a tableau there is a node $s \in \mathbf{S}$ such that $D \in \mathcal{L}'(s)$. A node $x \in V$ is created with $\pi(x) = s$ and $\mathcal{L}(x) = \{D\}$. Applications of $\sqsubseteq$-rule, $\sqcap$-rule, $\exists$-rule, $\sqcup$-rule, $\forall$-rule, $\forall_+$-rule, $\leq$-rule, $\geq$-rule and ch-rule preserve the conditions in (*). The proof is similar to that in [10]. It is not hard to check that applications of $\exists_+$-rule, $\oplus$-rule preserve the conditions in (*) as well.

We now show that if a completion tree $\mathbf{T}$ can be built with a function π satisfying (*) then $\mathbf{T}$ is clash-free.

1. If the condition 1 in (*) is satisfied then there is no node x in $\mathbf{T}$ such that $A, \dot{\neg}A \in \mathcal{L}(x)$ due to **P2** and the condition 1. That means that T does not contain a clash of the first kind as described in Definition 4.
2. There is no clash of the second kind in $\mathbf{T}$ if the conditions 1 to 3 in (*) are satisfied with **P12**.
3. Assume that $\exists Q^{\oplus}.C \in \mathcal{L}(x)$. Due to the condition 1 in (*), we have $\exists Q^{\oplus}.C \in \mathcal{L}'(\pi(x))$. According to **P8** and **P4**, there are $s_1, \cdots, s_n \in \mathbf{S}$ such that $\langle s_i, s_{i+1}\rangle \in \mathcal{E}(Q)$, $\exists Q^{\oplus}.C \in \mathcal{L}'(s_i)$ and $\{\exists Q.\exists Q^{\oplus}.C, \exists Q.C\} \cap \mathcal{L}'(s_i) \neq \emptyset$ for $i \in \{0, \cdots, n-1\}$ with $s_0 = \pi(x)$, and $\exists Q.C \in \mathcal{L}'(s) \cup \mathcal{L}'(s_{n-1})$.
   Assume $\exists Q.C \in \mathcal{L}'(s)$. Due to the condition 4 in (*), we have $\exists Q.C \in \mathcal{L}(x)$. This implies that $\mathbf{T}$ does not have a clash of the third kind.
   Assume $\exists Q.C \notin \mathcal{L}'(s)$ and $n > 1$. Without loss of the generality, assume that $\exists Q.C \notin \mathcal{L}'(s_i)$ for all $i \in \{0, \cdots, n-2\}$ and $\exists Q.C \in \mathcal{L}'(s_{n-1})$ (otherwise, if there is some $0 \leq k < n-1$ such that $\exists Q.C \in \mathcal{L}'(s_k)$ then we pick $n = k+1$).
   By applying successively $\exists$-rule, $\exists_+$-rule and $\sqcup$-rule, there are nodes $x_1, \cdots, x_l \in V$ such that $\pi(x_i) = s_i$, $Q \in \mathcal{L}(\langle x_{i-1}, x_i\rangle)$ and $\{\exists Q^{\oplus}.C, \exists Q.\exists Q^{\oplus}.C\} \subseteq \mathcal{L}(x_i)$ for all $i \leq l$ with some $l \leq n-1$. If $l = n-1$ then x has a $\exists Q^{\oplus}.C$-reachable node x_l such that $\exists Q.C \in \mathcal{L}(x_l)$ due to $\exists Q^{\oplus}.C \in \mathcal{L}(x_l)$, $\exists Q.C \in \mathcal{L}'(\pi(x_l))$ and the condition 4 in (*). If $l < n-1$ and x_l is blocked by z then we restart from the node z with $\exists Q^{\oplus}.C \in \mathcal{L}(z)$ (since $\mathcal{L}(z) = \mathcal{L}(x_l)$) finding $x'_1, \cdots, x'_{l'} \in V$ which have the same properties as those of $x_1, \cdots, x_l$. This process can be repeated until finding a node $w \in V$ such that w is a $\exists Q^{\oplus}.C$-reachable node of x.
   Therefore, $\mathbf{T}$ does not have a clash of the third kind. $\triangleleft$

The following theorem is a consequence of Lemmas 2, 3 and 4.

Theorem 1. *Algorithm 2 is a decision procedure for satisfiability of $\mathcal{SHIQ}_+$-concepts w.r.t. a $\mathcal{SHIQ}_+$ knowledge base.*

5 Conclusion and Discussion

We have presented a tableaux-based decision procedure for $\mathcal{SHIQ}_+$ concept satisfiability. In order to define tableaux for $\mathcal{SHIQ}_+$ we have introduced new properties that allow to represent semantic constraints imposed by transitive closure of roles and to avoid expressing explicitly cycles for role inclusion axioms with transitive closure. These new tableaux properties are translated into new non-deterministic expansion rules which cause the complexity of the tableaux-based algorithm presented in this paper to jump up to double exponential. An open issue consists in investigating whether this complexity is worst-case optimal. To the best of our knowledge, this problem has not addressed yet. Another future work concerns the extension of our tableaux-based algorithm to $\mathcal{SHIQ}_+$ with nominals.

Acknowledgements. Thanks to Ulrike Sattler for helpful discussions and to the anonymous reviewers for their comments.

References

1. Patel-Schneider, P., Hayes, P., Horrocks, I.: Owl web ontology language semantics and abstract syntax. In: W3C Recommendation (2004)
2. Tobies, S.: The complexity of reasoning with cardinality restrictions and nominals in expressive description logics. Journal of Artificial Intelligence Research 12, 199–217 (2000)
3. Le Duc, C.: Decidability of $\mathcal{SHI}$ with transitive closure of roles. 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. 368–383. Springer, Heidelberg (2009)
4. Le Duc, C., Lamolle, M.: Decidability of description logics with transitive closure of roles. In: Proceedings of the 23rd International Workshop on Description Logics (DL 2010), CEUR-WS.org (2010)
5. De Giacomo, G., Lenzerini, M.: Boosting the correspondence between description logics and propositional dynamic logics. In: Proceedings of the 12th National Conference on Artificial Intelligence, pp. 205–212. The MIT Press, Cambridge (1994)
6. De Giacomo, G., Lenzerini, M.: What's in an aggregate: Foundations for description logics with tuples and sets. In: Proceedings of the Fourteenth International Joint Conference On Intelligence Artificial 1995, IJCAI 1995 (1995)
7. Horrocks, I., Sattler, U.: Decidability of $\mathcal{SHIQ}$ with complex role inclusion axioms. Artificial Intelligence 160, 79–104 (2004)
8. Horrocks, I., Kutz, O., Sattler, U.: The even more irresistible $\mathcal{SROIQ}$. In: Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning. Springer, Heidelberg (2006)
9. Ortiz, M.: An automata-based algorithm for description logics around $\mathcal{SRIQ}$. In: Proceedings of the fourth Latin American Workshop on Non-Monotonic Reasoning 2008, CEUR-WS.org (2008)
10. Horrocks, I., Sattler, U., Tobies, S.: Practical reasoning for expressive description logics. In: Ganzinger, H., McAllester, D., Voronkov, A. (eds.) LPAR 1999. LNCS, vol. 1705. Springer, Heidelberg (1999)
11. Horrocks, I., Sattler, U.: A tableau decision procedure for $\mathcal{SHOIQ}$. Journal Of Automated Reasoning 39(3), 249–276 (2007)
12. Baader, F.: Augmenting concept languages by transitive closure of roles: An alternative to terminological cycles. In: Proceedings of the Twelfth International Joint Conference on Artificial Intelligence (1991)
13. Le Duc, C., Lamolle, M., Curé, O.: A tableaux-based algorithm for $\mathcal{SHIQ}$ with transitive closure of roles in concept and role inclusion axioms. In: Technical Report (2010), http://www.iut.univ-paris8.fr/files/webfm/recherche/linc/RR201012A.pdf

SPARQL Query Answering over OWL Ontologies

Ilianna Kollia[1,⋆], Birte Glimm[2], and Ian Horrocks[2]

[1] ECE School, National Technical University of Athens, Greece
[2] Oxford University Computing Laboratory, UK

Abstract. The SPARQL query language is currently being extended by W3C with so-called entailment regimes, which define how queries are evaluated under more expressive semantics than SPARQL's standard simple entailment. We describe a sound and complete algorithm for the OWL Direct Semantics entailment regime. The queries of the regime are very expressive since variables can occur within complex class expressions and can also bind to class or property names. We propose several novel optimizations such as strategies for determining a good query execution order, query rewriting techniques, and show how specialized OWL reasoning tasks and the class and property hierarchy can be used to reduce the query execution time. We provide a prototypical implementation and evaluate the efficiency of the proposed optimizations. For standard conjunctive queries our system performs comparably to already deployed systems. For complex queries an improvement of up to three orders of magnitude can be observed.

1 Introduction

Query answering is important in the context of the Semantic Web, since it provides a mechanism via which users and applications can interact with ontologies and data. Several query languages have been designed for this purpose, including RDQL, SeRQL and, most recently, SPARQL. In this paper, we consider the SPARQL [10] query language, which was standardized in 2008 by the World Wide Web Consortium (W3C) and which is now supported by most RDF triple stores. The query evaluation mechanism defined in the SPARQL Query specification [10] is based on subgraph matching. This form of query evaluation is also called simple entailment since it can equally be defined in terms of the simple entailment relation between RDF graphs. In order to use more elaborate entailment relations, such as those induced by RDF Schema (RDFS) or OWL semantics [4], SPARQL 1.1 includes several *entailment regimes*, including RDFS and OWL. Query answering under such entailment regimes is more complex as it may involve retrieving answers that only follow implicitly from the queried graph. While several methods and implementations for SPARQL under RDFS semantics are available, methods that use OWL semantics have not yet been well-studied.

For some of the less expressive OWL 2 profiles, an implementation of the entailment regime can make use of materialization techniques (e.g., for the OWL RL profile) or of query rewriting techniques (e.g., for the OWL QL profile). These techniques are, however, not applicable in general, and may not deal directly with all kinds of

⋆ Work done while at the Oxford University Computing Lab.

© Springer-Verlag Berlin Heidelberg 2011

SPARQL queries. In this paper, we present a sound and complete algorithm for answering SPARQL queries under the *OWL 2 Direct Semantics entailment regime* (from now on, SPARQL-OWL), describe a prototypical implementation based on the HermiT reasoner, and use this implementation to investigate a range of optimization techniques that improve query answering performance for different kinds of SPARQL-OWL queries.

The range of queries that can be formulated in SPARQL-OWL goes beyond standard conjunctive queries, which are already supported by several OWL reasoning systems. SPARQL-OWL does not allow for proper non-distinguished variables but it poses significant challenges for implementations, since, for example, variables can occur within complex class expressions and can also bind to class or property names. Amongst the query languages already supported by OWL reasoners, the closest in spirit to SPARQL-OWL is SPARQL-DL, which is implemented in the Pellet OWL reasoner [11]. SPARQL-DL is a subset of SPARQL-OWL that is designed such that queries can be mapped to standard reasoning tasks. In our algorithm, we extend the techniques used for conjunctive query answering to deal with arbitrary SPARQL-OWL queries and propose a range of novel optimizations in particular for SPARQL-OWL queries that go beyond SPARQL-DL.

We have implemented the optimized algorithm in a prototypical system, which is the first to fully support SPARQL-OWL, and we have performed a preliminary evaluation in order to investigate the feasibility of our algorithm and the effectiveness of the proposed optimizations. This evaluation suggests that, in the case of standard conjunctive queries, our system performs comparably to existing ones. It also shows that a naive implementation of our algorithm behaves badly for some non-standard queries, but that the proposed optimizations can dramatically improve performance, in some cases by as much as three orders of magnitude.

2 Preliminaries

We first give a brief introduction to OWL and RDF, followed by the definition of SPARQL's syntax and semantics and the SPARQL-OWL entailment regime. We generally abbreviate International Resource Identifiers (IRIs) using the prefixes rdf, rdfs, and owl to refer to the RDF, RDFS, and OWL namespaces, respectively. The empty prefix is used for an imaginary example namespace.

2.1 Web Ontology Language OWL

For OWL, we use the functional-style syntax (FSS), which directly reflects the OWL objects that are used to define the OWL 2 Direct Semantics. In the following subsection, we clarify how the OWL structural objects can be mapped into RDF triples. We present only several examples of typical OWL axioms; for a full definition of OWL 2, please refer to the OWL 2 Structural Specification and Direct Semantics [8,7].

$$\text{SubClassOf(:DogOwner ObjectSomeValuesFrom(:owns :Dog))} \tag{1}$$

$$\text{SubClassOf(:CatOwner ObjectSomeValuesFrom(:owns :Cat))} \tag{2}$$

$$\text{ObjectPropertyDomain(:owns :Person)} \tag{3}$$

ClassAssertion(ObjectUnionOf(:DogOwner :CatOwner) :mary) (4)

ObjectPropertyAssertion(:owns :mary _:somePet) (5)

Axioms (1) and (2) make use of existential quantification and state that every instance of the class :DogOwner (:CatOwner) is related to some instance of :Dog (:Cat) via the property :owns. Axiom (3) defines the domain of the property :owns as the class :Person, i.e., every individual that is related to some other individual with the :owns property belongs to the class :Person. Axiom (4) states that :mary belongs to the union of the classes :DogOwner and :CatOwner. Finally, Axiom (5) states that :mary owns some pet. The blank node _:somePet is called an *anonymous individual* in OWL and has an existential semantics. An OWL *ontology* contains a set of logical axioms, as the ones shown above, plus further non-logical statements, e.g., for the ontology header, type declarations (e.g., declaring :owns as an object property), or import directives. We focus on the logical axioms, which determine the logical consequences of the ontology.

More formally, the interpretation of axioms in an OWL ontology O is given by means of two-sorted interpretations over the *object domain* and the *data domain*, where the latter contains concrete values such as integers, strings, and so on. An *interpretation* maps classes to subsets of the object domain, object properties to pairs of elements from the object domain, data properties to pairs of elements where the first element is from the object domain and the second one is from the data domain, individuals to elements in the object domain, datatypes to subsets of the data domain, and literals (data values) to elements in the data domain. For an interpretation to be a *model* of an ontology, several conditions have to be satisfied [7]. For example, if O contains Axiom (4), then the interpretation of :mary must belong to the union of the interpretation of :DogOwner and :CatOwner. If an axiom ax is satisfied in every model of O, then we say that O *entails* ax, written $O \models ax$. For example, if O contains Axioms (1) to (5), then we have that O entails ClassAssertion(:Person :mary), i.e., we can infer that Mary is a person. This is because :mary will have an :owns-successor (due to Axiom (5)), which then implies that she belongs to the class :Person due to Axiom (3). In the same way, we say that an ontology O_1 entails another ontology O_2, written $O_1 \models O_2$, if every model of O_1 is also a model of O_2. The *vocabulary* Voc(O) of O is the set of all IRIs and literals that occur in O.

Note that the above axioms cannot be satisfied in a unique canonical model that could be used to answer queries since in one model we would have that :mary is a cat owner, whereas in another model, we would have that she is a dog owner. Thus, we cannot apply techniques such as forward chaining to materialize all consequences of the ontology. To satisfy the existential quantifiers (e.g, ObjectSomeValuesFrom), an OWL reasoner has to introduce new individuals, and in OWL it cannot be guaranteed that the models of an ontology are finite. OWL reasoners build, therefore, finite abstractions of models, which can be expanded into models.

2.2 Mapping to RDF Graphs

Since SPARQL is an RDF query language based on triples, we briefly show how the OWL objects introduced above can be mapped to RDF triples. The reverse direction, which maps triples to OWL objects is equally defined, but makes a well-formedness

Table 1. The RDF representation of Axioms (1), (3), and (4)

```
:DogOwner rdfs:subClassOf _:x .              :owns rdfs:domain :Person .      (3')
_:x rdf:type owl:restriction .               :mary rdf:type [ .
_:x owl:onProperty :owns .                     owl:unionOf
_:x owl:someValuesFrom :Dog .   (1')              (:DogOwner :CatOwner ) ] (4')
```

restriction, i.e., only certain RDF graphs can be mapped into OWL structural objects. We call such graphs *OWL 2 DL graphs*. For further details, we refer interested readers to the W3C specification that defines the mapping between OWL structural objects and RDF graphs [9].

Table 1 gives an RDF representation of Axioms (1), (3), and (4) in Turtle syntax [1]. OWL axioms that only use RDF Schema expressivity, e.g., domain and range restrictions, usually result in a straightforward translation. For example, Axiom (3) is mapped to the single triple (3'). Complex class expressions such as the super class in Axiom (1) usually require auxiliary blank nodes, e.g., we introduce the auxiliary blank node _:x for the superclass expression that is then used as the subject of subsequent triples. In the translation of Axiom (4), we further used Turtle's blank node constructor [ ] and ( ) as a shortcut for lists in RDF.

Note that it is now no longer obvious whether :owns is a data or an object property. This is why an RDF graph that represents an OWL DL ontology has to contain type declarations, i.e., although we did not show the type declarations in our example, we would expect to have a triple such as :owns a owl:ObjectProperty, which corresponds to the non-logical axiom Declaration(ObjectProperty(:owns)) in FSS.

2.3 Syntax and Semantics of SPARQL Queries

We do not recall the complete surface syntax of SPARQL here but simply introduce the underlying algebraic operations using our notation. A detailed introduction to the relationship of SPARQL queries and their algebra is given in [5].

SPARQL supports a variety of *filter expressions*, or just *filters*, built from RDF terms, variables, and a number of built-in functions and operators; see [10] for details.

Definition 1. *We write* I *for the set of all* IRIs*,* L *for the set of all* literals*, and* B *for the set of all* blank nodes*. The set* T *of* RDF terms *is* $\mathsf{I} \cup \mathsf{L} \cup \mathsf{B}$*. Let* V *be a countably infinite set of* variables *disjoint from* T*. A* triple pattern *is member of the set* $(\mathsf{T} \cup \mathsf{V}) \times (\mathsf{I} \cup \mathsf{V}) \times (\mathsf{T} \cup \mathsf{V})$*, and a* basic graph pattern *(BGP) is a set of triple patterns. More complex* graph patterns *are inductively defined to be of the form* BGP, $\mathsf{Join}(\mathsf{GP}_1, \mathsf{GP}_2)$, $\mathsf{Union}(\mathsf{GP}_1, \mathsf{GP}_2)$, $\mathsf{LeftJoin}(\mathsf{GP}_1, \mathsf{GP}_2, F)$*, and* $\mathsf{Filter}(\mathsf{F}, \mathsf{GP})$*, where* BGP *is a BGP,* F *is a filter, and* $\mathsf{GP}_{(i)}$ *are graph patterns that share no blank nodes.*[1] *The sets of* variables *and* blank nodes *in a graph pattern* GP *are denoted by* $\mathsf{V}(\mathsf{GP})$ *and* $\mathsf{B}(\mathsf{GP})$*, respectively.*

We exclude a number of SPARQL features from our discussion. First, we disregard any of the new SPARQL 1.1 query constructs since their syntax and semantics are still

[1] As in [10], disallowing GP_1 and GP_2 to share blank nodes is important to avoid unintended co-references.

under discussion in the SPARQL working group. Second, we do not consider output formats (e.g., SELECT or CONSTRUCT) and solution modifiers (e.g., LIMIT or OFFSET) which are not affected by entailment regimes. Third, we exclude SPARQL datasets that allow SPARQL endpoints to cluster data into several named graphs and a default graph. Consequently, we omit dataset clauses and assume that queries are evaluated over the default graph, called the *active graph* for the query.

Evaluating a SPARQL graph pattern results in a sequence of solutions that lists possible bindings of query variables to RDF terms in the active graph.

Definition 2. *A* solution mapping *is a partial function* $\mu\colon \mathsf{V} \to \mathsf{T}$ *from variables to RDF terms. For a solution mapping* μ *– and more generally for any (partial) function – the set of elements on which* μ *is defined is the* domain $\mathsf{dom}(\mu)$ *of* μ*, and the set* $\mathsf{ran}(\mu) := \{\mu(x) \mid x \in \mathsf{dom}(\mu)\}$ *is the* range *of* μ*. For a BGP* BGP*, we use* $\mu(\mathsf{BGP})$ *to denote the pattern obtained by applying* μ *to all elements of* BGP *in* $\mathsf{dom}(\mu)$*. Two solution mappings* μ_1 *and* μ_2 *are* compatible *if* $\mu_1(x) = \mu_2(x)$ *for all* $x \in \mathsf{dom}(\mu_1) \cap \mathsf{dom}(\mu_2)$*. If this is the case, a solution mapping* $\mu_1 \cup \mu_2$ *is defined by setting* $(\mu_1 \cup \mu_2)(x) = \mu_1(x)$ *if* $x \in \mathsf{dom}(\mu_1)$*, and* $(\mu_1 \cup \mu_2)(x) = \mu_2(x)$ *otherwise.*

This convention is extended in the obvious way to all functions that are defined on variables or terms.

Since SPARQL allows for repetitive solution mappings and since the order of solution mappings is only relevant for later processing steps, we use *solution multisets*. A *multiset* over an *underlying set* $S = \{s_1, \ldots, s_n\}$ is a set of pairs (s_i, m_i) with m_i a positive natural number, called the *multiplicity* of s_i.

We first define the evaluation of BGPs under SPARQL's standard semantics, which is also referred to as simple entailment or subgraph matching. We still need to consider, however, the effect of blank nodes in a BGP. Intuitively, these act like variables that are projected out of a query result, and thus they may lead to duplicate solution mappings. This is accounted for using RDF instance mappings as follows:

Definition 3. *An* RDF instance mapping *is a partial function* $\sigma\colon \mathsf{B} \to \mathsf{T}$ *from blank nodes to RDF terms. The* solution multiset *for a basic graph pattern* BGP *over the active graph* G *is the following multiset of solution mappings:*

$$\begin{aligned}\{(\mu, n) \mid\ & \mathsf{dom}(\mu) = \mathsf{V}(\mathsf{BGP}), \textit{and } n \textit{ is the maximal number such that}\\ & \sigma_1, \ldots, \sigma_n \textit{ are distinct RDF instance mappings with}\\ & \mathsf{dom}(\sigma_i) = \mathsf{B}(\mathsf{BGP}), \textit{for all } 1 \le i \le n, \textit{and } \mu(\sigma_i(\mathsf{BGP})) \textit{ is a subgraph of } \mathsf{G}\}.\end{aligned}$$

The algebraic operators that are required for evaluating non-basic graph patterns correspond to operations on multisets of solution mappings, which are the same for all entailment regimes. Thus, we refer interested readers to the SPARQL Query specification [10] or the work about entailment regimes in general [2].

2.4 SPARQL-OWL

The SPARQL-OWL entailment regime[2] specifies how the OWL Direct Semantics entailment relation can be used to evaluate BGPs of SPARQL queries. The regime assumes that the queried RDF graph G as well as the BGP are first mapped to OWL 2

[2] http://www.w3.org/TR/2010/WD-sparql11-entailment-20101014/

structural objects, which are extended to allow for variables. Graphs or BGPs that cannot be mapped since they are not well-formed, are rejected with an error. We use $\mathsf{O_G}$ to denote the result of mapping an OWL 2 DL graph G into an OWL ontology.

An *axiom template* is an OWL axiom, which can have variables in place of class, object property, data property, or individual names or literals. In order to map a BGP into a set of axiom templates, the entailment regime specification extends the mapping between RDF triples and structural OWL objects. Type declarations from $\mathsf{O_G}$ are used to disambiguate types in BGP, but the regime further requires type declarations for variables. This allows for a unique mapping from a BPG into axiom templates. For example, without variable typing, the BGP :mary ?pred ?obj could be mapped into a data or an object property assertion. By adding the triple ?pred a owl:ObjectProperty, we can uniquely map the BGP to an object property assertion. Given an OWL 2 DL graph G, we call BGP *well-formed w.r.t.* G if it can uniquely be mapped into axiom templates taking also the type declarations from $\mathsf{O_G}$ into account. We denote the resulting set of axiom templates with $\mathsf{O^G_{BGP}}$.

SPARQL's standard BGP evaluation trivially guarantees finite answers since it is based on subgraph matching. Since the entailment regimes use an entailment relation in the definition of BGP evaluation, infinite solution mappings that only vary in their use of different blank node labels have to be avoided. Thus, entailment regimes make use of Skolemization, which treats the blank nodes in the queried graph basically as constants, but ones that do not have any particular fixed name. Since Skolem constants should not occur in query results, Skolemization is only used to restrict the solution mappings.

Definition 4. *Let the prefix* skol *refer to a namespace IRI that does not occur as the prefix of any IRI in the active graph or query. The* Skolemization sk(_:b) *of a blank node* _:b *is defined as* sk(_:b) := skol:b. *With* $\mathsf{sk(O_G)}$ *we denote the result of replacing each blank node b in* $\mathsf{O_G}$ *with* sk(b). *Let* G *be an OWL 2 DL graph,* BGP *a BGP that is well-formed w.r.t.* G*, and* Voc(OWL) *the OWL vocabulary. The* answer domain w.r.t. G under OWL Direct Semantics entailment*, written* $\mathsf{AD_{DS}(G)}$*, is the set* $\mathsf{Voc(O_G)} \cup \mathsf{Voc(OWL)}$. *The evaluation of* $\mathsf{O^G_{BGP}}$ *over* $\mathsf{O_G}$ *under OWL 2 Direct Semantics entailment is defined as the solution multiset*

$$
\begin{aligned}
\{(\mu, n) \mid \mathsf{dom}(\mu) = \mathsf{V(BGP)}, & \textit{ and } n \textit{ is the maximal number such that} \\
& \sigma_1, \ldots, \sigma_n \textit{ are distinct RDF instance mappings such that, for each } 1 \le i \le n, \\
& i)\ \mathsf{O_G} \cup \mu(\sigma_i(\mathsf{O^G_{BGP}})) \textit{ is an OWL 2 DL ontology,} \\
& ii)\ \mathsf{sk(O_G)} \models \mathsf{sk}(\mu(\sigma_i(\mathsf{O^G_{BGP}}))) \textit{ and} \\
& iii)\ (\mathsf{ran}(\mu) \cup \mathsf{ran}(\sigma_i)) \subseteq \mathsf{AD_{DS}(G)}\}.
\end{aligned}
$$

Note that we only use Voc(OWL) for OWL's special class and property names such as owl:Thing or owl:TopObjectProperty.

3 SPARQL-OWL Query Answering

The SPARQL-OWL regime specifies what the answers are, but not how they can actually be computed. In this section, we describe an algorithm for SPARQL-OWL that internally uses any OWL 2 DL reasoner for checking entailment. We further describe optimizations that can be used to improve the performance of the algorithm by reducing the number of entailment checks and method calls to the reasoner. We assume that all axiom templates that are evaluated by our algorithm can be instantiated into logical axioms since non-logical axioms (e.g., type declarations) do not affect the consequences of an ontology. Since class variables can only be instantiated with class names, object property variables with object properties, etc., we first define which solution mappings are relevant for our algorithm.

Definition 5. *Let* G *be an OWL 2 DL graph and* BGP *a BGP that is well-formed w.r.t.* G. *By a slight abuse of notation, we write* $\mathsf{O}^{\mathsf{G}}_{\mathsf{BGP}} = \{\mathsf{axt}_1, \ldots, \mathsf{axt}_n\}$ *for* $\mathsf{axt}_1, \ldots, \mathsf{axt}_n$ *the logical axiom templates in* $\mathsf{O}^{\mathsf{G}}_{\mathsf{BGP}}$. *For* μ *a solution mapping and* σ *an RDF instance mapping, we call* (μ, σ) compatible with $\mathsf{O}^{\mathsf{G}}_{\mathsf{BGP}}$ and O_{G} *if* $\mu(\sigma(\mathsf{O}^{\mathsf{G}}_{\mathsf{BGP}}))$ *is such that (a)* $\mathsf{O}_{\mathsf{G}} \cup \mu(\sigma(\mathsf{O}^{\mathsf{G}}_{\mathsf{BGP}}))$ *is an OWL 2 DL ontology and (b)* $\mu(\sigma(\mathsf{O}^{\mathsf{G}}_{\mathsf{BGP}}))$ *is ground and does not contain fresh entities w.r.t.* $\mathsf{sk}(\mathsf{O}_{\mathsf{G}})$.

Condition (a) ensures that condition (i) of the entailment regime is satisfied, which guarantees that the OWL 2 DL constraints are not violated, e.g., only simple object properties can be used in cardinality constraints. Condition (b) makes sure that the variables are only instantiated with the corresponding types since otherwise we would introduce a fresh entity w.r.t. $\mathsf{sk}(\mathsf{O}_{\mathsf{G}})$ (e.g., by using an individual name as a class) or even violate the OWL 2 DL constraints. Furthermore, the condition ensures that $\mu(\sigma(\mathsf{O}^{\mathsf{G}}_{\mathsf{BGP}}))$ contains no blank nodes (it is ground) and is Skolemized since all entities in the range of μ and σ are from $\mathsf{sk}(\mathsf{O}_{\mathsf{G}})$. Thus, condition (iii) of entailment regimes holds.

Given an OWL 2 DL graph G and a well-formed BGP BGP for G, a straightforward algorithm to realize the entailment regime now maps G into O_{G}, BGP into $\mathsf{O}^{\mathsf{G}}_{\mathsf{BGP}}$, and then simply tests, for each compatible pair (μ, σ), whether $\mathsf{sk}(\mathsf{O}_{\mathsf{G}}) \models \mu(\sigma(\mathsf{O}^{\mathsf{G}}_{\mathsf{BGP}}))$. The notion of compatible solutions already reduces the number of possible solutions that have to be tested, but in the worst case, the number of distinct compatible pairs (μ, σ) is exponential in the number of variables in the query, i.e., if m is the number of terms in O_{G} and n is the number of variables in $\mathsf{O}^{\mathsf{G}}_{\mathsf{BGP}}$, we test $O(m^n)$ solutions. Such an algorithm is sound and complete if the reasoner used to decide entailment is sound and complete since we check all mappings for variables and blank nodes that can constitute actual solution and instance mappings.

3.1 General Query Evaluation Algorithm

Optimizations cannot easily be integrated in the above sketched algorithm since it uses the reasoner to check for the entailment of the instantiated ontology as a whole and, hence, does not take advantage of relations that may exist between axiom templates. For a more optimized BGP evaluation, we evaluate the BGP axiom template by axiom template. Initially, our solution set contains only the identity mapping, which does not map any variable or blank node to a value. We then pick our first axiom template, extend the identity mapping to cover the variables of the chosen axiom template and use the reasoner to check which of the mappings instantiate the axiom template into an entailed axiom. We then pick the next axiom template and again extend the mappings from the previous round to cover all variables and check which of those mappings lead to an entailed axiom. Thus, axiom templates which are very selective and are only satisfied by very few solutions reduce the number of intermediate solutions. Choosing a good execution order, therefore, can significantly affect the performance.

As an example, consider the BGP { ?x rdf:type :A . ?x :op ?y . } with :op an object property and :A a class. The query belongs to the class of conjunctive queries, i.e., we only query for class and property instances. We assume that the queried ontology contains 100 individuals, only 1 of which belongs to the class :A. This :A instance has 1 :op-successor, while we have overall 200 pairs of individuals related with the property :op. If we first evaluate ?x rdf:type :A (i.e., ClassAssertion(:A ?x)), we test 100 mappings (since x is an individual variable), of which only 1 mapping satisfies the axiom template. We then evaluate ?x :op ?y (i.e., ObjectPropertyAssertion(:op ?x ?y)) by extending the mapping with all 100 possible mappings for y. Again only 1 mapping yields a solution. For the reverse axiom template order, the first axiom template requires the test of $100 * 100$ mappings. Out of those, 200 remain to be checked for the second axiom template and we perform $10,200$ tests instead of just 200.

The importance of the execution order is well known in relational databases and cost based optimization techniques are used to find good execution orders. Ordering strategies as implemented in databases or triple stores are, however, not directly applicable in our setting. In the presence of expressive schema level axioms, we cannot rely on counting the number of occurrences of triples. We also cannot, in general, precompute all relevant inferences to base our statistics on materialized inferences. Furthermore, we should not only aim at decreasing the number of intermediate results, but also take into account the cost of checking or computing the solutions. This cost can be very significant with OWL reasoning.

Instead of checking entailment, we can, for several axiom templates, directly retrieve the solutions from the reasoner. For example, to evaluate a query with BGP { ?x rdfs:subClassOf :C }, which asks for subclasses of the class :C, we can use standard reasoner methods to retrieve the subclasses. Most methods of reasoners are highly optimized, which can significantly reduce the number of tests that are performed. Furthermore, if the class hierarchy is precomputed, the reasoner can find the answers simply with a cache lookup. Thus, the actual execution cost might vary significantly. Notably, we do not have a straight correlation between the number of results for an axiom template and the actual cost of retrieving the solutions as is typically the case in triple stores or databases. This requires cost models that take into account the cost of the specific reasoning operations (depending on the state of the reasoner) as well as the number of results.

As motivated above, we distinguish between *simple* and *complex* axiom templates, where simple axiom templates are those that correspond to dedicated reasoning tasks. Complex axiom templates are, in contrast, evaluated by iterating over the compatible mappings and by checking entailment for each instantiated axiom template. Examples of complex axiom templates are:

SubClassOf(:C ObjectIntersectionOf(?z ObjectSomeValuesFrom(?x ?y)))
ClassAssertion(ObjectSomeValuesFrom(:op ?x) ?y)

Algorithm 1 shows how we evaluate a BGP. The algorithm takes as input an OWL 2 DL graph G and basic graph pattern BGP that is well-formed w.r.t. G. It returns a multiset of solution mappings that is the result of evaluating BGP over G under the OWL 2

Algorithm 1. Query Evaluation Procedure

Input: G: the active graph, which is an OWL 2 DL graph
BGP: an OWL 2 DL BGP

Output: a multiset of solutions for evaluating BGP over G under OWL 2 Direct Semantics

```
1: $O_G$:=map(G)
2: $O^G_{BGP}$:=map(BGP, $O_G$)
3: Axt := rewrite($O^G_{BGP}$) {create a list Axt of simplified axiom templates from $O^G_{BGP}$}
4: $Axt^1, \ldots, Axt^m$:=connectedComponents(Axt)
5: for j=1, ..., m do
6:     $R_j := \{(\mu_0, \sigma_0) \mid dom(\mu_0) = dom(\sigma_0) = \emptyset\}$
7:     $axt_1, \ldots, axt_n$ := reorder($Axt^j$)
8:     for $i = 1, \ldots, n$ do
9:         $R_{new} := \emptyset$
10:        for $(\mu, \sigma) \in R_j$ do
11:            if isSimple($axt_i$) and $((V(axt_i) \cup B(axt_i)) \setminus (dom(\mu) \cup dom(\sigma))) \neq \emptyset$ then
12:                $R_{new} := R_{new} \cup \{(\mu \cup \mu', \sigma \cup \sigma') \mid (\mu', \sigma') \in callReasoner(\mu(\sigma(axt_i)))\}$
13:            else
14:                $B := \{(\mu \cup \mu', \sigma \cup \sigma') \mid dom(\mu') = V(\mu(axt_i)), dom(\sigma') = B(\sigma(axt_i)),$
                                       $(\mu \cup \mu', \sigma \cup \sigma')$ is compatible with $axt_i$ and $sk(O_G)\}$
15:                B := prune(B, $axt_i$, $O_G$)
16:                while $B \neq \emptyset$ do
17:                    $(\mu', \sigma')$ := removeNext($B$)
18:                    if $O_G \models \mu'(\sigma'(axt_i))$ then
19:                        $R_{new} := R_{new} \cup \{(\mu', \sigma')\}$
20:                    else
21:                        $B$ := prune($B$, $axt_i$, $(\mu', \sigma')$)
22:                    end if
23:                end while
24:            end if
25:        end for
26:        $R_j := R_{new}$
27:    end for
28: end for
29: $R := \{(\mu_1 \cup \ldots \cup \mu_m, \sigma_1 \cup \ldots \cup \sigma_m) \mid (\mu_j, \sigma_j) \in R_j, 1 \leq j \leq m\}$
30: return $\{(\mu, m) \mid m > 0$ is the maximal number with $\{(\mu, \sigma_1), \ldots, (\mu, \sigma_m)\} \subseteq R\}$
```

Direct Semantics. We first explain the general outline of the algorithm and leave the details of the used submethods for the following section. First, G and BGP are mapped to O_G and O^G_{BGP}, respectively (lines 1 and 2). The function rewrite (line 3) can be assumed to do nothing. Next, the method connectedComponents (line 4) partitions the axiom templates into sets of connected components, i.e., within a component the templates share common variables, whereas between components there are no shared variables. Unconnected components unnecessarily increase the amount of intermediate results and, instead, we can simply combine the results for the components in the end (line 29). For each component, we proceed as described below: we first determine an order (method reorder in line 7). For a simple axiom template, which contains so far unbound variables, we then call a specialized reasoner method to retrieve entailed results (callReasoner in line 12). Otherwise, we check which compatible solutions yield an entailed axiom (lines 13 to 24). The method prune can again be assumed do nothing.

3.2 Optimized Query Evaluation

Axiom Template Reordering We now explain how we order the axiom templates in the method reorder (line 7). Since complex axiom templates can only be evaluated with costly entailment checks, our aim is to reduce the number of bindings before we check the complex templates. Thus, we evaluate simple axiom templates first. The simple axiom templates are ordered by their cost, which is computed as the weighted sum of the estimated number of required consistency checks and the estimated result size. These estimates are based on statistics provided by the reasoner and this is the only part where our algorithm depends on the specific reasoner that is used. In case the reasoner cannot give estimates, one can still work with statistics computed from explicitly stated information and we do this for some simple templates, e.g., ObjectPropertyRange, for which the reasoner does not provide result size estimations. Since the result sizes for complex templates are difficult to estimate using either the reasoner or the explicitly stated information in O_G, we order complex templates based only on the number of bindings that have to be tested, i.e., the number of consistency checks that are needed to evaluate them. It is obvious that the reordering of axiom templates does not affect soundness and completeness of Algorithm 1.

Axiom Template Rewriting Some costly to evaluate axiom templates can be rewritten into axiom templates that can be evaluated more efficiently and yield an equivalent result. Such axiom templates are shown on the left-hand side of Table 2 and their equivalent simplified form is shown on the right-hand side. To understand the intuition behind such transformation, we consider a query with only the axiom template:

SubClassOf(?x ObjectIntersectionOf(ObjectSomeValuesFrom(:op ?y) :C))

This axiom template requires a quadratic number of consistency checks in the number of classes in the ontology (since ?x and ?y are class variables). According to Table 2, the rewriting yields:

SubClassOf(?x :C) and SubClassOf(?x ObjectSomeValuesFrom(:op ?y))

The first axiom template is now evaluated with a cheap cache lookup (assuming that the class hierarchy has been precomputed). For the second one, we only have to check the usually few resulting bindings for x combined with all other class names for y. For a complex axiom template such as the one in the last row of Table 2, the rewritten axiom template can be mapped to a specialized task of an OWL reasoner, which internally uses the class hierarchy to compute the domains and ranges with significantly fewer tests. We apply the rewriting from Table 2 in the method rewrite in line 3 of our algorithm. Our evaluation in Section 4 shows a significant reduction in running time due to this axiom template rewriting. Soundness and completeness is preserved since instantiated rewritten templates are semantically equivalent to the corresponding instantiated complex ones.

Table 2. Axiom templates and their equivalent simpler ones, where $C_{(i)}$ are class expressions (possibly containing variables), a is an individual or variable, and r is an object property expression (possibly containing a variable)

ClassAssertion(ObjectIntersectionOf(:C_1 ... :C_n) :a)	≡	{ClassAssertion(:C_i :a) | $1 \leq i \leq n$}
SubClassOf(:C ObjectIntersectionOf(:C_1 ... :C_n))	≡	{SubClassOf(:C :C_i) | $1 \leq i \leq n$}
SubClassOf(ObjectUnionOf(:C_1 ... :C_n) :C)	≡	{SubClassOf(:C_i :C) | $1 \leq i \leq n$}
SubClassOf(ObjectSomeValuesFrom(:op owl:Thing :C)	≡	ObjectPropertyDomain(:op :C)
SubClassOf(owl:Thing ObjectAllValuesFrom(:op :C))	≡	ObjectPropertyRange(:op :C)

Class-Property Hierarchy Exploitation The number of consistency checks needed to evaluate a BGP can be further reduced by taking the class and property hierarchies into account. Once the classes and properties are classified (this can ideally be done before a system accepts queries), the hierarchies are stored in the reasoner's internal structures. We further use the hierarchies to prune the search space of solutions in the evaluation of certain axiom templates. We illustrate the intuition with an example. Let us assume that O^G_{BGP} contains the axiom template:

SubClassOf(:Infection ObjectSomeValuesFrom(:hasCausalLinkTo ?x))

If :C is not a solution and SubClassOf(:B :C) holds, then :B is also not a solution. Thus, when searching for solutions for x, the method removeNext (line 17) chooses the next binding to test by traversing the class hierarchy topdown. When we find a non-solution :C, the subtree rooted in :C of the class hierarchy can safely be pruned, which we do in the method prune in line 21. Queries over ontologies with a large number of classes and a deep class hierarchy can, therefore, gain the maximum advantage from this optimization. We employ similar optimizations using the object and data property hierarchies. It is obvious that we only prune mappings that cannot constitute actual solution and instance mappings, hence, soundness and completeness of Algorithm 1 is preserved.

Exploiting the Domain and Range Restrictions Domain and range restrictions in O_G can be exploited to further restrict the mappings for class variables. Let us assume that O_G contains Axiom (6) and O^G_{BGP} contains Axiom Template (7).

ObjectPropertyRange(:takesCourse :Course) (6)

SubClassOf(:GraduateStudent ObjectSomeValuesFrom(:takesCourse ?x)) (7)

Only the class :Course and its subclasses can be solutions for x and we can immediately prune other mappings in the method prune (line 15), which again preserves soundness and completeness.

4 System Evaluation

Since entailment regimes only change the evaluation of basic graph patterns, standard SPARQL algebra processors can be used that allow for custom BGP evaluation. Furthermore, standard OWL reasoners can be used to perform the required reasoning tasks.

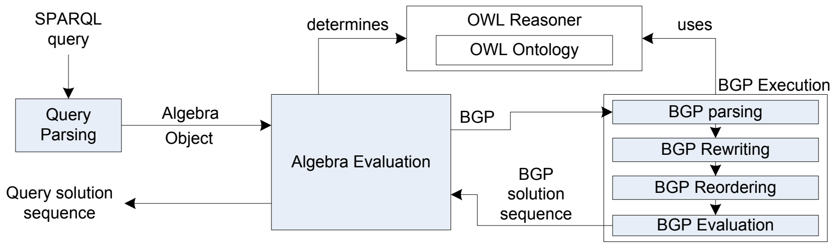


Fig. 1. The main phases of query processing in our system

4.1 The System Architecture

Figure 1 depicts the main phases of query processing in our prototypical system. In our setting, the queried graph is seen as an ontology that is loaded into an OWL reasoner. Currently, we only load the default graph/ontology of the RDF dataset into a reasoner and each query is evaluated using this reasoner. We plan, however, to extend the system to named graphs, where the dataset clause of the query can be used to determine a reasoner which contains one of the named ontologies instead of the default one. Loading the ontology and the initialization of the reasoner are performed before the system accepts queries. We use the ARQ library[3] of the Jena Semantic Web Toolkit for parsing the query and for the SPARQL algebra operations apart from our custom BGP evaluation method. The BGP is parsed and mapped into axiom templates by our extension of the OWL API [6], which uses the active ontology for type disambiguation. The resulting axiom templates are then passed to a query optimizer, which applies the axiom template rewriting and then searches for a good query execution plan based on statistics provided by the reasoner. We use the HermiT reasoner[4] for OWL reasoning, but only the module that generates statistics and provides cost estimations is HermiT specific.

4.2 Experimental Results

We tested our system with the Lehigh University Benchmark (LUBM) [3] and a range of custom queries that test complex axiom template evaluation over the more expressive GALEN ontology. All experiments were performed on a Windows Vista machine with a double core 2.2 GHz Intel x86 32 bit 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. Whether more costly operations such as the realization of the ABox, which computes the types for all individuals, are done in the beginning, depends on the setting and the reasoner. Since realization is relatively quick in HermiT for LUBM (GALEN has no individuals), we also performed this task upfront. The given results are averages from executing each query three times. The ontologies and all code required to perform the experiments are available online.[5]

[3] http://jena.sourceforge.net/ARQ/
[4] http://www.hermit-reasoner.com/
[5] http://www.hermit-reasoner.com/2010/sparqlowl/sparqlowl.zip

Table 3. Query answering times in milliseconds for LUBM(1,0) and in seconds for the queries of Table 4 with and without optimizations

LUMB(1, 0)	
Query	Time
1	20
2	46
3	19
4	19
5	32
6	58
7	42
8	353
9	4,475
10	23
11	19
12	28
13	16
14	45

GALEN queries from Table 4				
Query	Reordering	Hierarchy Exploitation	Rewriting	Time
1				2.1
1		x		0.1
2				780.6
2		x		4.4
3				>30 min
3		x		119.6
3			x	204.7
3		x	x	4.9
4	x		x	>30 min
4	x	x		361.9
4		x	x	>30 min
4	x	x	x	68.2
5	x			>30 min
5		x		>30 min
5	x	x		5.6

We first evaluate the 14 conjunctive ABox queries provided in the LUBM. These queries are simple ones and have variables only in place of individuals and literals. The LUBM ontology contains 43 classes, 25 object properties, and 7 data properties. We tested the queries on LUBM(1,0), which contains data for one university starting from index 0, and which contains 16,283 individuals and 8,839 literals. The ontology took 3.8 s to load and 22.7 s for classification and realization. Table 3 shows the execution time for each of the queries. The reordering optimization has the biggest impact on queries 2, 7, 8, and 9. These queries require much more time or are not answered at all within the time limit of 30 min without this optimization (758.9 s, 14.7 s, >30 min, >30 min, respectively).

Conjunctive queries are supported by a range of OWL reasoners. SPARQL-OWL allows, however, the creation of very powerful queries, which are not currently supported by any other system. In the absence of suitable standard benchmarks, we created a custom set of queries as shown in Table 4 (in FSS). Note that we omit variable type declarations since the variable types are unambiguous in FSS. Since the complex queries are mostly based on complex schema queries, we switched from the very simple LUBM ontology to the GALEN ontology. GALEN consists of 2,748 classes, 413 object properties, and no individuals or literals. The ontology took 1.6 s to load and 4.8 s to classify the classes and properties. The execution time for these queries is shown on the right-hand side of Table 3. For each query, we tested the execution once without optimizations and once for each combination of applicable optimizations from Section 3.

As expected, an increase in the number of variables within an axiom template leads to a significant increase in the query execution time because the number of mappings that have to be checked grows exponentially in the number of variables. This can, in particular, be observed from the difference in execution time between Query 1 and 2.

Table 4. Sample complex queries for the GALEN ontology

1	SubClassOf(:Infection ObjectSomeValuesFrom(:hasCausalLinkTo ?x))
2	SubClassOf(:Infection ObjectSomeValuesFrom(?y ?x))
3	SubClassOf(?x ObjectIntersectionOf(:Infection ObjectSomeValuesFrom(:hasCausalAgent ?y)))
4	SubClassOf(:NAMEDLigament ObjectIntersectionOf(:NAMEDInternalBodyPart ?x) SubClassOf(?x ObjectSomeValuesFrom(:hasShapeAnalagousTo ObjectIntersectionOf(?y ObjectSomeValuesFrom(?z :linear))))
5	SubClassOf(?x :NonNormalCondition) SubObjectPropertyOf(?z :ModifierAttribute) SubClassOf(:Bacterium ObjectSomeValuesFrom(?z ?w)) SubObjectProperty(?y :StatusAttribute) SubClassOf(?w :AbstractStatus) SubClassOf(?x ObjectSomeValuesFrom(?y :Status))

From Queries 1, 2, and 3 it is evident that the use of the hierarchy exploitation optimization leads to a decrease in execution time of up to two orders of magnitude and, in combination with the query rewriting optimization, we can get an improvement of up to three orders of magnitude as seen in Query 3. Query 4 can only be completed in the given time limit if at least reordering and hierarchy exploitation is enabled. Rewriting splits the first axiom template into the following two simple axiom templates, which are evaluated much more efficiently:

SubClassOf(NAMEDLigament NAMEDInternalBodyPart)
SubClassOf(NAMEDLigament ?x)

After the rewriting, the reordering optimization has an even more pronounced effect since both rewritten axiom templates can be evaluated with a simple cache lookup. Without reordering, the complex axiom template could be executed before the simple ones, which leads to the inability to answer the query within the time limit of 30 min. Without a good ordering, Query 5 can also not be answered, but the additional use of the class and property hierarchy further improves the execution time by three orders of magnitude.

Although our optimizations can significantly improve the query execution time, the required time can still be quite high. In practice, it is, therefore, advisable to add as many restrictive axiom templates for query variables as possible. For example, the addition of SubClassOf(?y Shape) to Query 4 reduces the runtime from 68.2 s to 1.6 s.

5 Discussion

We have presented a sound and complete query answering algorithm and novel optimizations for SPARQL's OWL Direct Semantics entailment regime. Our prototypical query answering system combines existing tools such as ARQ, the OWL API, and the HermiT OWL reasoner to implement an algorithm that evaluates basic graph patterns under OWL's Direct Semantics. Apart from the query reordering optimization—which uses (reasoner dependent) statistics provided by HermiT—the system is independent of the reasoner used, and could employ any reasoner that supports the OWL API.

We evaluated the algorithm and the proposed optimizations on the LUBM benchmark and on a custom benchmark that contains queries that make use of the very expressive features of the entailment regime. We showed that the optimizations can improve query execution time by up to three orders of magnitude.

Future work will include the creation of more accurate cost estimates for the cost-based query reordering, the implementation of caching strategies that reduce the number of tests for different instantiations of a complex axiom template, and an extended evaluation using a broader set of ontologies and queries. Finally, we plan to analyze whether user specific profiles can be used to suggest additional restrictive axiom templates automatically to reduce the number of mappings that have to be checked.

Acknowledgements. This work was supported by EPSRC in the project *HermiT: Reasoning with Large Ontologies*.

References

1. Beckett, D., Berners-Lee, T.: Turtle – Terse RDF Triple Language. W3C Team Submission (January 14, 2008), `http://www.w3.org/TeamSubmission/turtle/`
2. Glimm, B., Krötzsch, M.: SPARQL beyond subgraph matching. 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. 241–256. Springer, Heidelberg (2010)
3. Guo, Y., Pan, Z., Heflin, J.: LUBM: A benchmark for OWL knowledge base systems. J. Web Semantics 3(2-3), 158–182 (2005)
4. 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/`
5. Hitzler, P., Krötzsch, M., Rudolph, S.: Foundations of Semantic Web Technologies. Chapman & Hall/CRC (2009)
6. Horridge, M., Bechhofer, S.: The OWL API: A Java API for working with OWL 2 ontologies. In: Patel-Schneider, P.F., Hoekstra, R. (eds.) Proc. OWLED 2009 Workshop on OWL: Experiences and Directions. CEUR Workshop Proceedings, vol. 529, CEUR-WS.org (2009)
7. Motik, B., Patel-Schneider, P.F., Cuenca Grau, B. (eds.): OWL 2 Web Ontology Language: Direct Semantics. W3C Recommendation (October 27, 2009), `http://www.w3.org/TR/owl2-direct-semantics/`
8. 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/`
9. 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/`
10. Prud'hommeaux, E., Seaborne, A. (eds.): SPARQL Query Language for RDF. W3C Recommendation (January 15, 2008), `http://www.w3.org/TR/rdf-sparql-query/`
11. Sirin, E., Parsia, B.: SPARQL-DL: SPARQL query for OWL-DL. In: Golbreich, C., Kalyanpur, A., Parsia, B. (eds.) Proc. OWLED 2007 Workshop on OWL: Experiences and Directions. CEUR Workshop Proceedings, vol. 258, CEUR-WS.org (2007)

Epistemic Querying of OWL Knowledge Bases

Anees Mehdi, Sebastian Rudolph, and Stephan Grimm

Institute AIFB, Karlsruhe Institute of Technology, DE
{sebastian.rudolph,anees.mehdi}@kit.edu
Forschungszentrum Informatik Karlsruhe, DE
grimm@fzi.de

Abstract. Epistemic querying extends standard ontology inferencing by allowing for deductive introspection. We propose a technique for epistemic querying of OWL 2 ontologies not featuring nominals and universal roles by a reduction to a series of standard OWL 2 reasoning steps thereby enabling the deployment of off-the-shelf OWL 2 reasoning tools for this task. We prove formal correctness of our method, justify the omission of nominals and universal role, and provide an implementation as well as evaluation results.

1 Introduction

Ontologies play a crucial role in the Semantic Web and the Web Ontology Language (OWL, [10]) is the currently single most important formalism for web-based semantic applications. OWL 2 DL – the most comprehensive version of OWL that still allows for automated reasoning – is based on the description logic (DL) $\mathcal{SROIQ}$ [6]. Querying ontologies by means of checking entailment of axioms or instance retrieval is a crucial and prominent reasoning task in semantic applications. Despite being an expressive formalism, these standard querying capabilities with OWL ontologies lack the ability for introspection (i.e., asking what the knowledge base "knows" *within* the query language). Autoepistemic DLs cope with this problem and have been investigated in the context of OWL and Semantic Web. Particularly, they allow for introspection of the knowledge base in the query language via epistemic operators, such as the $\mathbf{K}$-operator (paraphrased as "known to be") that can be applied to concepts and roles.

The $\mathbf{K}$-operator allows for epistemic querying. E.g., in order to formulate queries like "known white wine that is not known to be produced in a French region" we could do an instance retrieval w.r.t. the DL concept $\mathbf{K}\,\mathit{WhiteWine} \sqcap \neg\exists\mathbf{K}\mathit{locatedIn}.\{\mathit{FrenchRegion}\}$. This can e.g. be used to query for wines that aren't explicitly excluded from being French wines but for which there is also no evidence of being French wines either (neither directly nor indirectly via deduction). For the knowledge base containing

$$\{\mathit{WhiteWine}(\mathit{MountadamRiesling}),\mathit{locatedIn}(\mathit{MountadamRiesling},\mathit{AustralianRegion})\}$$

the query would yield *MountadamRiesling* as a result, since it is known to be a white wine not known to be produced in France, while a similar query without epistemic operators would yield an empty result. Hence, in the spirit of

© Springer-Verlag Berlin Heidelberg 2011

nonmonotonicity, more instances can be retrieved (and thus conclusions can be drawn) than with conventional queries in this way. Another typical use case is integrity constraint checking: testing whether the axiom

$$\mathbf{K}\,\mathit{Wine} \sqsubseteq \exists\mathbf{K}\mathit{hasSugar}.\{\mathit{Dry}\} \sqcup \exists\mathbf{K}\mathit{hasSugar}.\{\mathit{OffDry}\} \sqcup \exists\mathbf{K}\mathit{hasSugar}.\{\mathit{Sweet}\}$$

is entailed allows to check whether for every named individual that is known to be a wine it is also known (i.e. it can be logically derived from the ontology) what degree of sugar it has.[1]

However, epistemic operators (or other means for nonmonotonicity) have not found their way into the OWL specification and current reasoners do not support this feature; former research has been focused on extending tableaux algorithms for less expressive formalisms than OWL and have not paced up with the development of OWL reasoners towards optimized tableaux for expressive languages; in particular, some expressive features like nominals require special care when combined with the idea of introspection by epistemic operators.

In this paper, we take a different approach to make epistemic querying possible with OWL ontologies; namely, we reuse existing OWL reasoners in a black box fashion while providing a mechanism for reducing the problem of epistemic querying to standard DL instance retrieval; our approach reduces occurrences of the **K**-operator to introspective look-ups of instances of a concept by calls to a standard DL reasoner, while we keep the number of such calls minimal; we have implemented this approach in form of a reasoner that accepts epistemic queries and operates on non-epistemic OWL ontologies Our contributions are the following:

– We introduce a transformation of epistemic queries to semantically identical non-epistemic queries by making introspective calls to a standard DL reasoner and by propagating the respective answer sets as nominals to the resulting query.
– We prove the correctness of this transformation in the light of some difficulties that occur with the common domain and rigid term assumptions that underly autoepistemic DLs.
– We present an efficient algorithm for implementing the above transformation with a minimal number of calls to a standard DL reasoner for the introspective look-ups of instances.
– Based on this algorithm, we provide a reasoner capable of answering epistemic queries by means of reduction to standard DL reasoning in the framework of the OWL-API extended by constructs for epistemic concepts and roles to be used in epistemic queries. First experiments show that our approach to epistemic querying is practically feasible.

The rest of this paper is structured as follows: Section 2 puts our approach into context with related work. Section 3 introduces the description logic $\mathcal{SROIQ}$

[1] Note that this cannot be taken for granted even if $\mathit{Wine} \sqsubseteq \exists\mathit{hasSugar}.\{\mathit{Dry}\} \sqcup \exists\mathit{hasSugar}.\{\mathit{OffDry}\} \sqcup \exists\mathit{hasSugar}.\{\mathit{Sweet}\}$ is stated in (or can be derived from) the ontology.

and its extension with the epistemic operator $\mathbf{K}$. In Section 4, we provide the formal justification for our method of reducing $\mathcal{SROIQK}$ axiom entailment from $\mathcal{SRIQ}$ knowledge bases. In Section 5, we describe principle problems arising from allowing the use of nominals or universal role in the knowledge base. In Section 6, we discuss the implementation issues and some evaluation results. We conclude in Section 7. For details and proofs we refer to the accompanying technical report [8].

2 Related Work

In the early 80s, H. J. Levesque expressed the need for a richer query language in knowledge formalisms [7]. He argues that the approach to knowledge representation should be functional rather than structural and defends the idea of extending a query language by the operator *knows* denoted by $\mathbf{K}$. In [11], Raymond Reiter makes a similar argument of in-adequacy of the standard first-order language for querying. Nevertheless, he discusses this issue in the context of databases. Similar lines of argumentation can be seen in the DL-community as well [4,5,3,2] where several extensions of DLs have been presented as well as algorithms for deciding the diverse reasoning tasks in such extensions. The extension of the DL $\mathcal{ALC}$ [12] by the epistemic operator $\mathbf{K}$ called $\mathcal{ALCK}$, is presented in [4]. A tableau algorithm has been designed for deciding the satisfiability problem. Answering queries in $\mathcal{ALCK}$ put to $\mathcal{ALC}$ knowledge bases is also discussed. In [9], a hybrid formalism is presented which integrates DLs and rules and also captures epistemic querying to DL knowledge bases. In this work we mainly focus on DLs extended with the epistemic operator $\mathbf{K}$ following notions presented in [4]. However, we consider more expressive DLs rather than just $\mathcal{ALC}$. Our approach is also in the spirit of [1] as it exploits a correspondence between epistemic querying and iterated non-epistemic querying.

3 Preliminaries

We present an introduction to the description logic $\mathcal{SROIQ}$ and its extension with the epistemic operator $\mathbf{K}$. Let N_I, N_C, and N_R be finite, disjoint sets called *individual names*, *concept names* and *role names* respectively, with $N_R = \mathbf{R_s} \uplus \mathbf{R_n}$ called *simple* and *non-simple* roles, respectively. These atomic entities can be used to form complex ones in the usual way (see Table 1). A *$\mathcal{SROIQ}$-knowledge base* is a tuple $(\mathcal{T}, \mathcal{R}, \mathcal{A})$ where $\mathcal{T}$ is a $\mathcal{SROIQ}$-TBox, $\mathcal{R}$ is a regular $\mathcal{SROIQ}$-role hierarchy[2] and $\mathcal{A}$ is a $\mathcal{SROIQ}$-ABox containing axioms as presented in Table 2. The semantics of $\mathcal{SROIQ}$ is defined via interpretations $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$ composed of a non-empty set $\Delta^\mathcal{I}$ called the *domain of* $\mathcal{I}$ and a function $\cdot^\mathcal{I}$ mapping individuals to elements of $\Delta^\mathcal{I}$, concepts to subsets of $\Delta^\mathcal{I}$ and roles to subsets of $\Delta^\mathcal{I} \times \Delta^\mathcal{I}$. This mapping is extended to complex roles and concepts as in Table 1 and finally used to evaluate axioms (see Table 2). We say $\mathcal{I}$ satisfies

[2] We assume the usual regularity assumption for $\mathcal{SROIQ}$, but omit it for space reasons.

Table 1. Syntax and semantics of role and concept constructors in $\mathcal{SROIQ}$. Thereby a denotes an individual name, R an arbitrary role name and S a simple role name. C and D denote concept expressions.

Name	Syntax	Semantics
inverse role	R^-	$\{\langle x,y\rangle \in \Delta^\mathcal{I} \times \Delta^\mathcal{I} \mid \langle y,x\rangle \in R^\mathcal{I}\}$
universal role	U	$\Delta^\mathcal{I} \times \Delta^\mathcal{I}$
top	$\top$	$\Delta^\mathcal{I}$
bottom	$\bot$	$\emptyset$
negation	$\neg C$	$\Delta^\mathcal{I} \setminus C^\mathcal{I}$
conjunction	$C \sqcap D$	$C^\mathcal{I} \cap D^\mathcal{I}$
disjunction	$C \sqcup D$	$C^\mathcal{I} \cup D^\mathcal{I}$
nominals	$\{a\}$	$\{a^\mathcal{I}\}$
univ. restriction	$\forall R.C$	$\{x \in \Delta^\mathcal{I} \mid \langle x,y\rangle \in R^\mathcal{I} \text{ implies } y \in C^\mathcal{I}\}$
exist. restriction	$\exists R.C$	$\{x \in \Delta^\mathcal{I} \mid \text{for some } y \in \Delta^\mathcal{I}, \langle x,y\rangle \in R^\mathcal{I} \text{ and } y \in C^\mathcal{I}\}$
Self concept	$\exists S.\mathsf{Self}$	$\{x \in \Delta^\mathcal{I} \mid \langle x,x\rangle \in S^\mathcal{I}\}$
qualified number	$\leqslant n\, S.C$	$\{x \in \Delta^\mathcal{I} \mid \#\{y \in \Delta^\mathcal{I} \mid \langle x,y\rangle \in S^\mathcal{I} \text{ and } y \in C^\mathcal{I}\} \leq n\}$
restriction	$\geqslant n\, S.C$	$\{x \in \Delta^\mathcal{I} \mid \#\{y \in \Delta^\mathcal{I} \mid \langle x,y\rangle \in S^\mathcal{I} \text{ and } y \in C^\mathcal{I}\} \geq n\}$

Table 2. Syntax and semantics of $\mathcal{SROIQ}$ axioms

Axiom α	$\mathcal{I} \models \alpha$, if	
$R_1 \circ \cdots \circ R_n \sqsubseteq R$	$R_1^\mathcal{I} \circ \cdots \circ R_n^\mathcal{I} \subseteq R^\mathcal{I}$	RBox $\mathcal{R}$
$\mathsf{Dis}(S,T)$	$S^\mathcal{I} \cap T^\mathcal{I} = \emptyset$	
$C \sqsubseteq D$	$C^\mathcal{I} \subseteq D^\mathcal{I}$	TBox $\mathcal{T}$
$C(a)$	$a^\mathcal{I} \in C^\mathcal{I}$	ABox $\mathcal{A}$
$R(a,b)$	$(a^\mathcal{I}, b^\mathcal{I}) \in R^\mathcal{I}$	
$a \doteq b$	$a^\mathcal{I} = a^\mathcal{I}$	
$a \not\doteq b$	$a^\mathcal{I} \neq b^\mathcal{I}$	

a knowledge base $\Sigma = (\mathcal{T}, \mathcal{R}, \mathcal{A})$ (or $\mathcal{I}$ is a model of Σ, written: $\mathcal{I} \models \Sigma$) if it satisfies all axioms of $\mathcal{T}$, $\mathcal{R}$, and $\mathcal{A}$. We say that a knowledge base Σ *entails* an axiom α (written $\Sigma \models \alpha$) if all models of Σ are models of α.

Next, we present the extension of the DL $\mathcal{SROIQ}$ by the epistemic operator K. Let $\mathcal{SROIQK}$ denote the extension of $\mathcal{SROIQ}$ by K, where we allow K to appear in front of concept or role expressions. We call a $\mathcal{SROIQK}$-role an *epistemic role* if $\mathbf{K}$ occurs in it. An epistemic role is *simple* if it is of the form $\mathbf{K}S$ where S is a simple $\mathcal{SROIQ}$-role.

The semantics of $\mathcal{SROIQK}$ is given as *possible world semantics* in terms of *epistemic interpretations*. Thereby the following two central assumptions are made:

1. *Common Domain Assumption*: all interpretations are defined over a fixed infinite domain Δ.
2. *Rigid Term Assumption*: For all interpretations, the mapping from individuals to domains elements is fixed: it is just the identity function.

Definition 1. An *epistemic interpretation* for $\mathcal{SROIQK}$ is a pair $(\mathcal{I}, \mathcal{W})$ where $\mathcal{I}$ is a $\mathcal{SROIQ}$-interpretation and $\mathcal{W}$ is a set of $\mathcal{SROIQ}$-interpretations, where $\mathcal{I}$ and all of $\mathcal{W}$ have the same infinite domain Δ with $N_I \subset \Delta$. The interpretation function $\cdot^{\mathcal{I},\mathcal{W}}$ is then defined as follows:

$$
\begin{aligned}
a^{\mathcal{I},\mathcal{W}} &= a \quad \text{for } a \in N_I \\
X^{\mathcal{I},\mathcal{W}} &= X^{\mathcal{I}} \quad \text{for } A \in N_C \cup N_R \cup \{\top, \bot\} \\
(\mathbf{K}C)^{\mathcal{I},\mathcal{W}} &= \textstyle\bigcap_{\mathcal{J}\in\mathcal{W}}(C^{\mathcal{J},\mathcal{W}}) \qquad (\mathbf{K}R)^{\mathcal{I},\mathcal{W}} = \textstyle\bigcap_{\mathcal{J}\in\mathcal{W}}(R^{\mathcal{J},\mathcal{W}}) \\
(C \sqcap D)^{\mathcal{I},\mathcal{W}} &= C^{\mathcal{I},\mathcal{W}} \cap D^{\mathcal{I},\mathcal{W}} \qquad (C \sqcup D)^{\mathcal{I},\mathcal{W}} = C^{\mathcal{I},\mathcal{W}} \cup D^{\mathcal{I},\mathcal{W}} \\
(\neg C)^{\mathcal{I},\mathcal{W}} &= \Delta \setminus C^{\mathcal{I},\mathcal{W}} \\
(\exists R.\mathsf{Self})^{\mathcal{I},\mathcal{W}} &= \{p \in \Delta \mid (p,p) \in R^{\mathcal{I},\mathcal{W}}\} \\
(\exists R.C)^{\mathcal{I},\mathcal{W}} &= \{p_1 \in \Delta \mid \exists p_2.(p_1,p_2) \in R^{\mathcal{I},\mathcal{W}} \wedge p_2 \in C^{\mathcal{I},\mathcal{W}}\} \\
(\forall R.C)^{\mathcal{I},\mathcal{W}} &= \{p_1 \in \Delta \mid \forall p_2.(p_1,p_2) \in R^{\mathcal{I},\mathcal{W}} \rightarrow p_2 \in C^{\mathcal{I},\mathcal{W}}\} \\
(\leqslant nR.C)^{\mathcal{I},\mathcal{W}} &= \{d \mid \#\{e \in C^{\mathcal{I},\mathcal{W}} \mid (d,e) \in R^{\mathcal{I},\mathcal{W}}\} \leq n\} \\
(\geqslant nR.C)^{\mathcal{I},\mathcal{W}} &= \{d \mid \#\{e \in C^{\mathcal{I},\mathcal{W}} \mid (d,e) \in R^{\mathcal{I},\mathcal{W}}\} \geq n\}
\end{aligned}
$$

where C and D are $\mathcal{SROIQK}$-concepts and R is a $\mathcal{SROIQK}$-role. $\Diamond$

From the above one can see that $\mathbf{K}C$ is interpreted as the set of objects that are in the interpretation of C under every interpretation in $\mathcal{W}$. Note that the rigid term assumption implies the unique name assumption (UNA) i.e., for any epistemic interpretation $\mathcal{I} \in \mathcal{W}$ and for any two distinct individual names a and b we have that $a^{\mathcal{I}} \neq b^{\mathcal{I}}$.

The notions of GCI, assertion, role hierarchy, ABox, TBox and knowledge base, and their interpretations as defined for $\mathcal{SROIQ}$ can be extended to $\mathcal{SROIQK}$ in the obvious way.

An *epistemic model* for a $\mathcal{SROIQK}$-knowledge base $\Sigma = (\mathcal{T}, \mathcal{R}, \mathcal{A})$ is a *maximal* non-empty set $\mathcal{W}$ of $\mathcal{SROIQ}$-interpretations such that $(\mathcal{I}, \mathcal{W})$ satisfies $\mathcal{T}$, $\mathcal{R}$ and $\mathcal{A}$ for each $\mathcal{I} \in \mathcal{W}$. A $\mathcal{SROIQK}$-knowledge base Σ is said to be *satisfiable* if it has an epistemic model. The knowledge base Σ *(epistemically) entails* an axiom α (written $\Sigma \models\!\!\!| \alpha$), if for every epistemic model $\mathcal{W}$ of Σ, we have that for every $\mathcal{I} \in \mathcal{W}$, the epistemic interpretation $(\mathcal{I}, \mathcal{W})$ satisfies α. By definition every $\mathcal{SROIQ}$-knowledge base is an $\mathcal{SROIQK}$-knowledge base. Note that a given $\mathcal{SROIQ}$-knowledge base Σ has up to isomorphism only one unique epistemic model which is the set of all models of Σ having infinite domain and satisfying the unique name assumption. We denote this model by $\mathcal{M}(\Sigma)$.

4 Deciding Entailment of Epistemic Axioms

In this section we provide a way for deciding epistemic entailment based on techniques for non-epistemic standard reasoning. More precisely, we consider the problem whether a $\mathcal{SROIQK}$ axiom α is entailed by a $\mathcal{SRIQ}$ knowledge base Σ, where $\mathcal{SRIQ}$ is defined as $\mathcal{SROIQ}$ excluding nominals and the universal role. That is, we distinguish the *querying language* from the *modeling language.* One primary use of the $\mathbf{K}$ operator that we focus on in this paper is for knowledge base introspection in the query, which justifies to exclude it from the modeling language in exchange for reducibility to standard reasoning.

The reasons for disallowing the use of nominals and the universal role will be discussed in Section 5.

The basic, rather straightforward idea to decide entailment of an axiom containing **K** operators is to disassemble the axiom, query for the named individuals contained in extensions for every subexpression preceded by **K**, and use the results to rewrite the axiom into one that is free of **K**s. While we will show that this idea is theoretically and practically feasible, some problems need to be overcome that arise from the definition of epistemic models, in particular the rigid term assumption and the common domain assumption.

As a consequence of the rigid name assumption, every $\mathcal{I} \in \mathcal{M}(\Sigma)$ satisfies the condition that individual names are interpreted by different individuals (this condition per se is commonly referred to as the *unique* name assumption). In order to enforce this behavior (which is not ensured by the non-epistemic standard DL semantics) we have to explicitly axiomatize this condition.

Definition 2. Given a $\mathcal{SRIQ}$ knowledge base Σ, we denote by Σ_{UNA} the knowledge base $\Sigma \cup \{a \not\doteq b \mid a, b \in N_I, a \neq b\}$. $\Diamond$

Fact 3. *The set of models of Σ_{UNA} is exactly the set of those models of Σ that satisfy the unique name assumption.*

As another additional constraint on epistemic interpretations, the domain is required to be infinite (imposed by the common domain assumption). However, standard DL reasoning as performed by OWL inference engines adheres to a semantics that allows for both finite and infinite models. Therefore, in order to show that we can use standard inferencing tools as a basis of epistemic reasoning, we have to prove that finite models can be safely dismissed from the consideration, without changing the results. We obtain this result by arguing that for any finite interpretation we find an infinite one which "behaves the same" in terms of satisfaction of axioms and hence will make up for the loss of the former. The following definition and lemma provide a concrete construction for this.

Definition 4. For any $\mathcal{SRIQ}$ interpretation $\mathcal{I}$, the *lifting* of $\mathcal{I}$ to ω is the interpretation $\mathcal{I}_\omega$ defined as follows:
- $\Delta^{\mathcal{I}_\omega} := \Delta^{\mathcal{I}} \times \mathbb{N}$,
- $a^{\mathcal{I}_\omega} := \langle a^{\mathcal{I}}, 0 \rangle$ for every $a \in N_I$,
- $A^{\mathcal{I}_\omega} := \{\langle x, i \rangle \mid x \in A^{\mathcal{I}} \text{ and } i \in \mathbb{N}\}$ for all $A \in N_C$,
- $r^{\mathcal{I}_\omega} := \{(\langle x, i \rangle, \langle x', i \rangle) \mid (x, x') \in r^{\mathcal{I}} \text{ and } i \in \mathbb{N}\}$ for all $r \in N_R$. $\Diamond$

Lemma 5. *Let Σ be a $\mathcal{SRIQ}$ knowledge base. For any interpretation $\mathcal{I}$ we have that $\mathcal{I} \models \Sigma$ if and only if $\mathcal{I}_\omega \models \Sigma$.*

The actual justification for our technique of rewriting axioms containing **K**s into **K**-free ones exploiting intermediate reasoner calls comes from the fact that (except for some remarkable special cases) the semantic extension of expressions proceeded by **K** can only contain named individuals. We prove this by exploiting certain symmetries on the model set $\mathcal{M}(\Sigma)$. Intuitively, one can freely swap or permute anonymous individuals (i.e., domain elements which do not correspond to any individual name) in a model of some knowledge base without losing modelhood, as detailed in the following definition and lemma.

Definition 6. Given an interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, a set Δ with $N_I \subseteq \Delta$, and a bijection $\varphi : \Delta^{\mathcal{I}} \to \Delta$ with $\varphi(a^{\mathcal{I}}) = a$ for all $a \in N_I$, the *renaming* of $\mathcal{I}$ according to φ, denoted by $\varphi(\mathcal{I})$, is defined as the interpretation $(\Delta, \cdot^{\varphi(\mathcal{I})})$ with

- $a^{\varphi(\mathcal{I})} = \varphi(a^{\mathcal{I}}) = a$ for every individual name a,
- $A^{\varphi(\mathcal{I})} = \{\varphi(z) \mid z \in A^{\mathcal{I}}\}$ for every concept name A, and
- $P^{\varphi(\mathcal{I})} = \{(\varphi(z), \varphi(w)) \mid (z, w) \in P^{\mathcal{I}}\}$ for every role name P. ◊

Lemma 7. *Let Σ be a $\mathcal{SRIQ}$ knowledge base and let $\mathcal{I}$ be a model of Σ with infinite domain. Then, every renaming $\varphi(\mathcal{I})$ of $\mathcal{I}$ satisfies $\varphi(\mathcal{I}) \in \mathcal{M}(\Sigma)$.*

Proof. By definition, the renaming satisfies the common domain and rigid term assumption. Modelhood w.r.t. Σ immediately follows from the isomorphism lemma of first-order interpretations [13] since $\mathcal{I}$ and $\varphi(\mathcal{I})$ are isomorphic and φ is an isomorphism from $\mathcal{I}$ to $\varphi(\mathcal{I})$. □

This insight can be used to "move" every anonymous individual into the position of another individual which serves as a counterexample for membership in some given concept D, unless the concept is equivalent to $\top$. This allows to prove that $\mathbf{K}D$ contains merely named individuals, given that it is not universal.

Lemma 8. *Let Σ be a $\mathcal{SRIQ}$ knowledge base. For any epistemic concept $C = \mathbf{K}D$ with $\Sigma_{\text{UNA}} \not\models D \equiv \top$ and $x \in \Delta$, we have that $x \in C^{\mathcal{I},\mathcal{M}(\Sigma)}$ iff x is named such that there is an individual $a \in N_I$ with $x = a^{\mathcal{I},\mathcal{M}(\Sigma)}$ and $\Sigma_{\text{UNA}} \models D(a)$.*

A similar property can be proved for the roles as well. Before, we have to take care of the exceptional case of the universal role.

Claim 9. *Let Σ be a knowledge base. For the universal role U we have: $\mathbf{K}U^{\mathcal{I},\mathcal{M}(\Sigma)} = U^{\mathcal{I},\mathcal{M}(\Sigma)}$*

The claim follows trivially as $U^{\mathcal{J}} = \Delta \times \Delta$ for any $\mathcal{J} \in \mathcal{M}(\Sigma)$. This means that $\bigcap_{\mathcal{J} \in \mathcal{M}(\Sigma)} U^{\mathcal{J}} = \Delta \times \Delta$. Thus, as in the case of concepts, whenever an epistemic concept contains a role of the form $\mathbf{K}U$, it will be simply replaced by U. That, for $\mathcal{SRIQ}$ knowledge bases, no other role than U is universal (in all models) is straightforward and can be shown using the construction from Definition 4.

We can now also show that the extension of every role preceded by $\mathbf{K}$ (except for the universal one), consists only of pairs of named individuals.

Lemma 10. *Let Σ be a $\mathcal{SRIQ}$ knowledge base. For any epistemic role $R = \mathbf{K}P$ with $P \neq U$, and $x, y \in \Delta$ we have that $(x, y) \in R^{\mathcal{I},\mathcal{M}(\Sigma)}$ iff at least one of the following holds:*

1. *there are individual names $a, b \in N_I$ such that $a^{\mathcal{I},\mathcal{M}(\Sigma)} = x$, $b^{\mathcal{I},\mathcal{M}(\Sigma)} = y$ and $\Sigma_{\text{UNA}} \models P(a, b)$.*
2. *$x = y$ and $\Sigma_{\text{UNA}} \models \top \sqsubseteq \exists P.\mathit{Self}$.*

Having established the above correspondences, we are able to define a translation procedure that maps (complex) epistemic concept expressions to non-epistemic ones which are equivalent in all models of Σ.

Definition 11. Given a $\mathcal{SRIQ}$ knowledge base Σ, we define the function Φ_Σ mapping $\mathcal{SROIQK}$ concept expressions to $\mathcal{SROIQ}$ concept expressions as follows (where we let $\{\} = \emptyset = \bot$)[3]:

$$\Phi_\Sigma : \left\{ \begin{array}{rl} C \mapsto & C \quad \text{if } C \text{ is an atomic or one-of concept, } \top \text{ or } \bot; \\ \mathbf{K}D \mapsto & \begin{cases} \top & \text{if } \Sigma_{\text{UNA}} \models \Phi_\Sigma(D) \equiv \top \\ \{a \in N_I \mid \Sigma_{\text{UNA}} \models \Phi_\Sigma(D)(a)\} & \text{otherwise} \end{cases} \\ \exists \mathbf{K}S.\mathsf{Self} \mapsto & \begin{cases} \exists S.\mathsf{Self} & \text{if } \Sigma_{\text{UNA}} \models \top \sqsubseteq \exists S.\mathsf{Self} \\ \{a \in N_I \mid \Sigma_{\text{UNA}} \models S(a,a)\} & \text{otherwise} \end{cases} \\ C_1 \sqcap C_2 \mapsto & \Phi_\Sigma(C_1) \sqcap \Phi_\Sigma(C_2) \\ C_1 \sqcup C_2 \mapsto & \Phi_\Sigma(C_1) \sqcup \Phi_\Sigma(C_2) \\ \neg C \mapsto & \neg\Phi_\Sigma(C) \\ \exists R.D \mapsto & \exists R.\Phi_\Sigma(D) \quad \text{for non-epistemic role } R \\ \exists \mathbf{K}P.D \mapsto & \bigsqcup_{a \in N_I} \{a\} \sqcap \exists P.(\{b \in N_I \mid \Sigma_{\text{UNA}} \models P(a,b)\} \sqcap \Phi_\Sigma(D)) \\ & \quad \sqcup \begin{cases} \Phi_\Sigma(D) & \text{if } \Sigma_{\text{UNA}} \models \top \sqsubseteq \exists P.\mathsf{Self} \\ \bot & \text{otherwise} \end{cases} \\ \forall R.D \mapsto & \forall R.\Phi_\Sigma(D) \quad \text{for non-epistemic role } R; \\ \forall \mathbf{K}P.D \mapsto & \neg\Phi_\Sigma(\exists \mathbf{K}P.\neg D) \\ \geqslant nS.D \mapsto & \geqslant nS.\Phi_\Sigma(D) \quad \text{for non-epistemic role } S; \\ \geqslant n\mathbf{K}S.D \mapsto & \begin{cases} \bigsqcup_{a \in N_I} \{a\} \sqcap \geqslant nP.(\{b \in N_I \mid \Sigma_{\text{UNA}} \models P(a,b)\} \sqcap \Phi_\Sigma(D)) & \text{if } n{>}1 \\ \Phi_\Sigma(\exists \mathbf{K}P.D) & \text{otherwise} \end{cases} \\ \leqslant nS.D \mapsto & \leqslant nS.\Phi_\Sigma(D) \quad \text{for non-epistemic role } S; \\ \leqslant n\mathbf{K}S.D \mapsto & \neg\Phi_\Sigma(\geqslant (n{+}1)\mathbf{K}S.D) \\ \Xi\mathbf{K}U.D \mapsto & \Xi U.\Phi_\Sigma(D) \quad \text{for } \Xi \in \{\forall, \exists, \geqslant n, \leqslant n\} \end{array} \right.$$

◊

We are now ready to establish the correctness of this translation in terms of (epistemic) entailment. In the following lemma, we show that the extension of a $\mathcal{SROIQK}$-concept and the extension of the $\mathcal{SROIQ}$-concept, obtained using the translation function Φ_Σ, agree under each model of the knowledge base.

Lemma 12. *Let Σ be a $\mathcal{SRIQ}$-knowledge base, x be an element of Δ, and C be a $\mathcal{SROIQK}$ concept. Then for any interpretation $\mathcal{I} \in \mathcal{M}(\Sigma)$, we have that $C^{\mathcal{I},\mathcal{M}(\Sigma)} = (\Phi_\Sigma(C))^{\mathcal{I},\mathcal{M}(\Sigma)}$.*

Moreover Lemma 12 allows to establish the result that the translation function Φ_Σ can be used to reduces the problem of entailment of $\mathcal{SROIQK}$ axioms by $\mathcal{SRIQ}$ knowledge bases to the problem of entailment of $\mathcal{SROIQ}$ axioms, formally put into the following theorem.

Theorem 13. *For a $\mathcal{SRIQ}$ knowledge base Σ, $\mathcal{SROIQK}$-concepts C and D and an individual a the following hold:*

1. *$\Sigma \mathrel{|}\!\models C(a)$ exactly if $\Sigma_{\text{UNA}} \models \Phi_\Sigma(C)(a)$.*
2. *$\Sigma \mathrel{|}\!\models C \sqsubseteq D$ exactly if $\Sigma_{\text{UNA}} \models \Phi_\Sigma(C) \sqsubseteq \Phi_\Sigma(D)$.*

[3] W.l.o.g. we assume that in the definition of Φ_Σ, $n \geq 1$.

Proof. For the first case, we see that $\Sigma \Vdash C(a)$ is equivalent to $a^{\mathcal{I},\mathcal{M}(\Sigma)} \in C^{\mathcal{I},\mathcal{M}(\Sigma)}$ which by Lemma 12 is the case exactly if $a^{\mathcal{I},\mathcal{M}(\Sigma)} \in \Phi_\Sigma(C)^{\mathcal{I},\mathcal{M}(\Sigma)}$ for all $\mathcal{I} \in \mathcal{M}(\Sigma)$. Since $\Phi_\Sigma(C)$ does not contain any **K**s, this is equivalent to $a^\mathcal{I} \in \Phi_\Sigma(C)^\mathcal{I}$ and hence to $\mathcal{I} \models \Phi_\Sigma(C)(a)$ for all $\mathcal{I} \in \mathcal{M}(\Sigma)$. Now we can invoke Fact 3 and Lemma 5 to see that this is the case if and only if $\Sigma_{\text{UNA}} \models \Phi_\Sigma(C)(a)$. The second case is proven in exactly the same fashion. □

Hence standard DL-reasoners can be used in order to answer epistemic queries. It can be seen from the definition of Φ_Σ that deciding epistemic entailment along those lines may require deciding many classical entailment problems and hence involve many calls to the reasoner. Nevertheless, the number of reasoner calls is bounded by the number of **K**s occurring in the query.

5 Semantical Problems Caused by Nominals and the Universal Role

One of the basic assumptions that is made regarding the epistemic interpretations is the *common domain assumption* as mentioned in Section 3. It basically has two parts: all the interpretations considered in an epistemic interpretation share the same fixed domain and the domain is infinite. However, there is no prima facie reason, why the domain that is described by a knowledge base should not be finite, yet finite models are excluded from the consideration entirely. We have shown that this is still tolerable for description logics up to $\mathcal{SRIQ}$ due to the fact that every finite model of a knowledge base gives rise to an infinite one that behaves the same (i.e. the two models cannot be distinguished by means of the underlying logic), as shown in Lemma 5. However, this situation changes once nominals or the universal role are allowed. In fact, the axioms $\top \sqsubseteq \{a, b, c\}$ or $\top \sqsubseteq \leqslant 3U.\top$ have only models with at most three elements. Consequently, according to the prevailing epistemic semantics, these axioms are epistemically unsatisfiable. In general, the coincidence of $\Vdash$ and $\models$ under the UNA which holds for nonepistemic KBs and axioms up to $\mathcal{SRIQ}$ does not hold any more, once nominals or the universal role come into play.

We believe that this phenomenon is not intended but rather a side effect of a semantics crafted for and probed against less expressive description logics, as it contradicts the intuition behind the **K** operator. A refinement of the semantics in order to ensure an intuitive behavior also in the presence of very expressive modeling features is subject of ongoing research.

6 A System

To check the feasibility of our method in practice, we have implemented a system that we called *EQuIKa*[4] and performed some first experiments for epistemic querying.

[4] **E**pistemic **Qu**erying **I**nterfance **Ka**rlsruhe.

Algorithm 1. translate (Σ, C) – Translate epistemic query concepts to non-epistemic ones

Require: a $\mathcal{SRIQ}$ knowledge base Σ, an epistemic concept C
Ensure: the return value is the non-epistemic concept $\Phi_\Sigma(C)$

```
translate ($\Sigma$, $C = \mathbf{K}D$)
  $\mathcal{X}$ := retrieveInstances ($\Sigma$, translate ($\Sigma$,$D$))
  **return** $\{\ldots, o_i, \ldots\}$    , $o_i \in \mathcal{X}$
translate ($\Sigma$, $C = \exists\mathbf{K}R.D$)
  $\mathcal{X} := \bot$
  $\mathcal{X}_D$ := retrieveInstances ($\Sigma$, translate ($\Sigma$,$D$))
  for each $a \in N_I$
    $\mathcal{X}_R$ := retrieveInstances ($\Sigma$, $\exists.R^-\{a\}$)
    $\mathcal{X} := \mathcal{X} \sqcup (\{a\} \sqcap \exists R.(\mathcal{X}_R \sqcap \mathcal{X}_D))$
  if $\Sigma \models \top \sqsubseteq \exists R.\mathsf{Self}$
    $\mathcal{X} := \mathcal{X} \sqcup \mathcal{X}_D$
  **return** $\mathcal{X}$
translate ($\Sigma$, $C = \forall\mathbf{K}R.D$)
  $\mathcal{X}_{\bar{D}}$ := retrieveInstances ($\Sigma$, translate ($\Sigma$,$\neg D$))
  $\mathcal{X}$ := retrieveInstances ($\Sigma$, $\exists R.\{\ldots, o_i, \ldots\}$)    , $o_i \in \mathcal{X}_{\bar{D}}$
  **return** $\neg\{\ldots, o_i, \ldots\}$    , $o_i \in \mathcal{X}$
translate ($\Sigma$, $C = \ldots$)
  ...
```

Implementation: The *EQuIKa* system implements the transformation of an epistemic concept to its non-epistemic version from Definition 11 involving calls to an underlying standard DL reasoner that offers the reasoning task of instance retrieval. To obtain an efficient implementation of Φ_Σ it is crucial to keep the number of calls to the DL reasoner minimal. With Algorithm 1 we provide such an efficient implementation, exploiting the fact that the extension of an epistemic role P (that occur in role restrictions) only contains pairs of known individuals provided neither $P = U$ nor $\Sigma \models \top \sqsubseteq \exists P.\mathsf{Self}$ is the case. It shows the transformation in terms of recursive translation functions for the various cases of epistemic concept expressions.

An important point, as far as optimization is concerned, is to reduced the number of calls to the underlying DL reasoner. From Algorithm 1, it can be seen that the number of calls to the underlying DL reasoner is at most twice the number of $\mathbf{K}$-operators that occur in the original query. This is much better than a naive implementation of Φ_Σ according to Definition 11 with iteration over intermediate retrieved individuals.

The *EQuIKa* system is implemented on top of the OWL-API[5] extending its classes and interfaces with constructs for epistemic concepts and roles, as shown by the UML class diagram in Figure 1. The new types OWLObjectEpistemicConcept and OWLObjectEpistemicRole are derived from the respective standard types OWLBooleanClassExpression and OWLObjectPropertyExpression to fit the design of the OWL-API.

[5] http://owlapi.sourceforge.net/

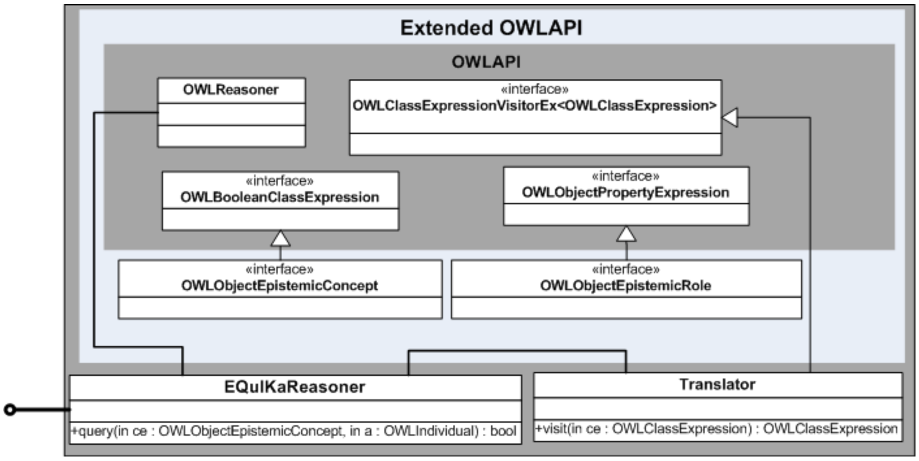

Extended OWLAPI
OWLAPI
OWLReasoner
«interface»
OWLClassExpressionVisitorEx<OWLClassExpression>
«interface»
OWLBooleanClassExpression
«interface»
OWLObjectPropertyExpression
«interface»
OWLObjectEpistemicConcept
«interface»
OWLObjectEpistemicRole
EQuIKaReasoner
+query(in ce : OWLObjectEpistemicConcept, in a : OWLIndividual) : bool
Translator
+visit(in ce : OWLClassExpression) : OWLClassExpression

Fig. 1. The *EQuIKa*-system extending the OWL-API

Using these types, the transformation Φ_Σ is implemented in the class Translator following the visitor pattern mechanism built in the OWL-API, which is indicated by the virtual translation functions with different arguments in Algorithm 1. Finally, the EQuIKaReasoner uses both a Translator together with an OWLReasoner to perform epistemic reasoning tasks.

Experiments: For the purpose of testing, we chose two versions of the wine ontology[6] with 483 and 1127 instances. As a measure, we took the time required to translate an epistemic concept to a non-epistemic equivalent one and the instance retrieval time of the translated concept. This suffices as entailment check can not be harder than instance retrieval. We investigate different epistemic concepts. For each such concept C, we consider a non-epistemic concept obtained from C by dropping the K-operators from it (see Table 3). Given a concept C, $\mathsf{t}_{(C)}$ and $|C_i|$ represent the time in seconds required to compute the instances and the number of instances computed for C_i. Finally for an epistemic concept EC_i, $\mathsf{t}_{\mathsf{T}(EC_i)}$ represents the time required by EQuIKa to translate EC_i to its non-epistemic equivalent. Table 4 provides our evaluation results. One can see from the evaluation results in Table 4 that the time required to compute the number of instances is feasible; it is roughly in the same order of magnitude as for non-epistemic concepts. Note also that the runtime comparison between epistemic concepts EC_i and their non-epistemic counterparts C_i should be taken with a grain of salt as they are semantically different in general, as also indicated by the fact that there are cases where retrieval for the epistemic concept takes less time than for the non-epistemic version. As a general observation, we noticed that instances retrieval for an epistemic concept where a K-operator occurs within the scope of a negation, tends to require much time.

[6] `http://www.w3.org/TR/owl-guide/wine.rdf`

Table 3. Concepts used for instance retrieval experiments

C_1	$\exists hasWineDescriptor.WineDescriptor$
EC_1	$\exists \mathsf{K} hasWineDescriptor.\mathsf{K}\, WineDescriptor$
C_2	$\forall hasWineDescriptor.WineDescriptor$
EC_2	$\forall \mathsf{K} hasWineDescriptor.\mathsf{K}\, WineDescriptor$
C_3	$\exists hasWineDescriptor.WineDescriptor \sqcap \exists madeFromFruit.WineGrape$
EC_3	$\exists \mathsf{K} hasWineDescriptor.\mathsf{K}\, WineDescriptor \sqcap \exists \mathsf{K} madeFromFruit.\mathsf{K}\, WineGrape$
C_4	$WhiteWine \sqcap \neg\exists locatedIn.\{FrenchRegion\}$
EC_4	$\mathsf{K}\, WhiteWine \sqcap \neg\exists \mathsf{K} locatedIn.\{FrenchRegion\}$
C_5	$Wine \sqcap \neg\exists hasSugar.\{Dry\} \sqcap \neg\exists hasSugar.\{OffDry\} \sqcap \neg\exists hasSugar.\{Sweet\}$
EC_5	$\mathsf{K}\, Wine \sqcap \neg\exists \mathsf{K} hasSugar.\{Dry\} \sqcap \neg\exists \mathsf{K} hasSugar.\{OffDry\} \sqcap \neg \mathsf{K}\exists hasSugar.\{Sweet\}$

Table 4. Evaluation

Ontology	Concept	$\mathsf{t}_{(C_i)}$	$\lvert C_i \rvert$	Concept	$\mathsf{t}_{\mathsf{T}(EC_i)}$	$\mathsf{t}_{(EC_i)}$	$\lvert EC_i \rvert$
Wine 1	C_1	2.13	159	EC_1	46.98	0.04	3
	C_2	0.01	483	EC_2	0.18	0.00	0
	C_3	28.90	159	EC_3	79.43	6.52	3
	C_4	0.13	0	EC_4	95.60	107.82	72
	C_5	52.23	80	EC_5	60.78	330.49	119
Wine 2	C_1	8.51	371	EC_1	351.78	0.13	308
	C_2	0.30	1127	EC_2	0.127	0.00	0
	C_3	227.10	371	EC_3	641.24	19.58	7
	C_4	0.34	0	EC_4	865.04	840.97	168
	C_5	295.87	240	EC_5	381.41	2417.65	331

7 Conclusion

In this work, we have introduced a way to answer epistemic queries to restricted OWL 2 DL ontologies via a reduction to a series of standard reasoning steps. This enables the deployment of today's highly optimized OWL inference engines for this non-standard type of queries. Experiments have shown that the approach is computationally feasible with runtimes in the same order of magnitude as standard (non-epistemic) reasoning tasks.

We identify the following avenues for future research: first and foremost we want to extend the expressivity of the underlying knowledge base to full OWL 2 DL, including nominals and the universal role. To this end, we have to alter the semantics and relinquishing the common domain assumption, to retain an intuitive entailment behavior. Second, we will provide a language extension to OWL 2 for epistemic operators in order to provide for a coherent way of serializing epistemic axioms. Finally we will investigate to which extent the promoted blackbox approach can be extended to the case where the epistemic operator occurs inside the considered knowledge base – note however, that in this case there is no unique epistemic model anymore.

Acknowledgement

This work is supported by the Deutsche Forschungsgemeinschaft (DFG) under the ExpresST project.

References

1. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Eql-lite: Effective first-order query processing in description logics. In: Proc. of the 20th Int. Joint Conf. on Artificial Intelligence (IJCAI 2007), pp. 274–279 (2007)
2. Donini, F., Nardi, D., Rosati, R.: Non-first-order features in concept languages. In: Proceedings of the Fourth Conference of the Italian Association for Artificial Intelligence (AI*IA 1995). LNCS (LNAI), pp. 91–102. Springer, Heidelberg (1995)
3. Donini, F.M., Lenzerini, M., Nardi, D., Schaerf, A., Nutt, W.: Adding epistemic operators to concept languages. In: Nebel, B., Rich, C., Swartout, W.R. (eds.) Proceedings of the 3rd International Conference on Principles of Knowledge Representation and Reasoning, pp. 342–353. Morgan Kaufmann, San Francisco (1992)
4. Donini, F.M., Lenzerini, M., Nardi, D., Schaerf, A., Nutt, W.: An epistemic operator for description logics. Artificial Intelligence 100(1-2), 225–274 (1998)
5. Donini, F.M., Nardi, D., Rosati, R.: Autoepistemic description logics. In: Proc. of IJCAI 1997, pp. 136–141 (1997)
6. Horrocks, I., Kutz, O., Sattler, U.: The even more irresistible $\mathcal{SROIQ}$. In: Doherty, P., Mylopoulos, J., Welty, C.A. (eds.) Proceedings of the 10th International Conference on Principles of Knowledge Representation and Reasoning (KR 2006), pp. 57–67. AAAI Press, Menlo Park (2006)
7. Levesque, H.J.: Foundations of a functional approach to knowledge representation. Artif. Intell. 23(2), 155–212 (1984)
8. Mehdi, A., Rudolph, S., Grimm, S.: Epistemic queries for owl. Technical report, Institut AIFB, KIT, Karlsruhe (December 2010), http://www.aifb.kit.edu/web/Techreport3009
9. Motik, B., Rosati, R.: Reconciling description logics and rules. Journal of the ACM 57(5) (2010)
10. W3C OWL Working Group. OWL 2 Web Ontology Language: Document Overview. W3C Recommendation, October 27 (2009), http://www.w3.org/TR/owl2-overview/
11. Reiter, R.: What should a database know? J. Log. Program. 14(1-2), 127–153 (1992)
12. Schmidt-Schauß, M., Smolka, G.: Attributive concept descriptions with complements. Artif. Intell. 48(1), 1–26 (1991)
13. van Dalen, D.: Logic and Structure. Springer, Heidelberg (1989)

A Practical Approach for Computing Generalization Inferences in $\mathcal{EL}$

Rafael Peñaloza and Anni-Yasmin Turhan

TU Dresden, Germany
Institute of Theoretical Computer Science
last name@tcs.inf.tu-dresden.de

Abstract. We present methods that compute generalizations of concepts or individuals described in ontologies written in the Description Logic $\mathcal{EL}$. These generalizations are the basis of methods for ontology design and are the core of concept similarity measures. The reasoning service least common subsumer (lcs) generalizes a set of concepts. Similarly, the most specific concept (msc) generalizes an individual into a concept description. For $\mathcal{EL}$ with general $\mathcal{EL}$-TBoxes, the lcs and the msc may not exist. However, it is possible to find a concept description that is the lcs (msc) up to a certain role-depth.

In this paper we present a practical approach for computing the lcs and msc with a bounded depth, based on the polynomial-time completion algorithm for $\mathcal{EL}$ and describe its implementation.

1 Introduction

Ontologies have become a commonly used means to describe controlled vocabularies, most prominently, in life sciences. Categories that form these vocabularies are sometimes only described in terms of specializations, i.e. by the "is-a" relation. Since the standardization of the web ontology language OWL [25], more applications have begun using this richer modeling language for describing notions from their domain in a more precise and detailed way. The formalism underlying OWL are Description Logics (DLs) [3], which are a family of logics with formal semantics. The formal semantics of DLs are the basis for the definition of reasoning services such as *subsumption* or *instance checking*. Subsumption tests whether a sub- / super-concept relationship holds between a pair of concept descriptions. Instance checking answers the question whether it follows from the ontology that a given individual must belong to a concept. The reasoning algorithms for these reasoning services are well-investigated for a range of DLs and implemented in powerful reasoner systems. In this paper we want to devise computation methods for inferences that can be employed to derive generalizations. These inferences turn out to be useful for range of ontology-based applications such as e.g. the life sciences [21,9] or context-aware systems [22].

The newest version of the OWL standard [25] offers several *OWL profiles*, which correspond to DLs with varying expressivity. We are interested in the OWL EL profile, which corresponds to the DL $\mathcal{EL}$++, an extension of the DL $\mathcal{EL}$ where reasoning is still tractable. $\mathcal{EL}$-concept descriptions are composed from conjunctions or existential restrictions. Despite its limited expressivity, $\mathcal{EL}$ has turned out to be useful to model

© Springer-Verlag Berlin Heidelberg 2011

notions from life science applications. Most prominently, the medical ontology SnoMed [21] and the Gene Ontology [9] are written in $\mathcal{EL}$. For instance, it is possible to express by

$$\mathsf{Myocarditis} \sqsubseteq \mathsf{inflammation} \sqcap \exists \mathsf{has{-}location.heart}$$

that myocarditis is a kind of inflammation that is located in the heart.

In fact, medical and context-aware applications deal with very large ontologies, which are often *light-weight*, in the sense that they can be formulated in $\mathcal{EL}$ or one of its extensions from the so-called $\mathcal{EL}$-family. Members of the $\mathcal{EL}$-family allow for reasoning in polynomial time [2]. In particular, subsumption and instance checking are tractable for $\mathcal{EL}$ and $\mathcal{EL}$++, which was the main reason to standardize it in an own OWL 2 profile [25]. The reasoning algorithms for the $\mathcal{EL}$-family are based on a completion method and have been implemented in optimized reasoners such as CEL [16].

We investigate here two inferences that generalize different entities from DL knowledge bases. The first one is the *least common subsumer* (lcs) [7], which generalizes a collection of concept descriptions into a single concept description that is the least w.r.t. subsumption. Intuitively, the lcs yields a new (complex) concept description that captures all the commonalities of the input concept descriptions. The second inference is the *most specific concept* (msc) [4], which generalizes an individual into a concept description. Intuitively, the msc delivers the most specific concept description that is capable of describing the individual.

Applying Generalization Inferences

In the following we describe some of the most prominent applications of the lcs and the msc.

Similarity measures. Concept similarity measures compute, given a pair of concept descriptions, a numerical value between 0 and 1 that lies closer to 1 the more similar the concepts are. Similarity measures are an important means to discover, for instance, functional similarities of genes modeled in ontologies. In [13] and, more recently, in [19] several similarity measures were evaluated for the Gene Ontology and it was concluded that the similarity measure from Resnik [20] performed well, if not best. This similarity measure is an edge-based approach, which finds the most specific common ancestor (msa) [1] of the concepts to be compared in the concept hierarchy and computes a similarity value based on the number of edges between the concepts in question and their msa. Clearly, the msa can only yield a named concept from the TBox and thus captures possibly only *some* of the commonalities of the concepts to be compared. The lcs, in contrast, captures *all* commonalities and is thus a more faithful starting point for a similarity measure. In fact, the lcs was employed for similarity measures for DLs in [6] already. In a similar fashion a similarity measure for comparing individuals can be based on the msc [10].

Building ontologies. In [11] it was observed that users working with biological ontologies would like to develop the description of the application categories in an example-driven way. More precisely, users would like to start by modeling individuals which are

[1] Sometimes also called *least common ancestor* (lca)

then generalized into a concept description. In fact, in the bottom-up approach for the construction of knowledge bases [4], a collection of individuals is selected for which a new concept definition is to be introduced in the ontology. Such a definition can be generated automatically by first generalizing each selected individual into a concept description (by computing the msc for each of them) and then applying the lcs to these concept descriptions.

The lcs can also be employed to enrich unbalanced concept hierarchies by adding new intermediate concepts [23].

Reconciling heterogeneous sources. The bottom-up procedure sketched before can also be employed in applications that face the problem that different information sources provide differing observations for the same state of affairs. For instance, in context-aware systems a GPS sensor or a video camera can provide differing information on a the location of a user. Alternatively, in medical applications, different diagnosing methods may yield differing results. It can be determined what the different sources agree on by representing this information as distinct ABox individuals and then by finding a common generalization of them by the bottom-up approach.

Information retrieval. The msc inference can be employed to obtain a query concept from an individual to search for other, similar individuals in an ontology [15,8].

In order to support all these ontology services for practical applications automatically, computation algorithms for the generalization inferences in $\mathcal{EL}$ are needed. Unfortunately, the lcs in $\mathcal{EL}$ does not always exist, when computed w.r.t. cyclic TBoxes [1]. Similarly, the msc in $\mathcal{EL}$ does not always exist, if the ABox is cyclic [12], mainly because cyclic structures cannot be captured in $\mathcal{EL}$-concept descriptions. In [12] the authors propose to use an approximation of the msc by limiting the role-depth of the concept description computed. We pursue this approach here for the lcs and the msc and thus would obtain only “common subsumers” and “specific concepts” that are still generalizations of the input, but not necessarily the least ones w.r.t. subsumption. However, by our proposed method we obtain *the* lcs or *the* msc w.r.t. the given role depth bound. We argue that such approximations are still useful in practice.

Recently, a different approach for obtaining the lcs (or the msc) in presence of cyclic knowledge bases was proposed in [14] by extending $\mathcal{EL}$ with concept constructors for greatest fixpoints. In the so obtained DL $\mathcal{EL}^{\nu}$ reasoning stays polynomial and the lcs and msc w.r.t. cyclic knowledge bases can be computed. However, the DL obtained by adding constructors for greatest fixpoints is possibly not easy to comprehend for naive users of ontologies.

For medical or context-aware applications knowledge bases can typically grow very large in practice. Thus, in order to support the computation of the (role-depth bounded) lsc or the msc for such applications, efficient computation of these generalizations for $\mathcal{EL}$ is desirable. Our computation methods build directly on the completion method for subsumption and instance checking for $\mathcal{EL}$ [2] for which optimizations already exists and are employed in modern reasoner systems. This enables the implementation of the role-depth bounded lcs and msc on top of existing reasoner systems. More precisely, in our completion-based approach, we obtain the role-depth bounded lcs by traversing the data-structures built during the computation of the subsumption hierarchy of the ontology. The role-depth bounded msc can be obtained from the data-structures generated during the computation of all instance relations for the knowledge base. We have recently implemented the completion-based computation of the role-depth bounded lcs and msc in our system GEL.

This paper is structured as follows: after introducing basic notions of DLs, we discuss the completion algorithms for classification and instance checking in $\mathcal{EL}$ in Section 3. We extend these methods to computation algorithms for the role-depth bounded lcs in Section 4.1 and for the role-depth bounded msc in Section 4.2 and we describe our initial implementation of the presented methods in Section 5. We conclude the paper with an outline of possible future work.

2 Preliminaries

We now formally introduce the DL $\mathcal{EL}$. Let N_I, N_C and N_R be disjoint sets of *individual names*, *concept names* and *role names*, respectively. *$\mathcal{EL}$-concept descriptions* are built according to the syntax rule

$$C ::= \top \mid A \mid C \sqcap D \mid \exists r.C$$

where $A \in N_C$, and $r \in N_R$.

A *general concept inclusion* (GCI) is a statement of the form $C \sqsubseteq D$, where C, D are $\mathcal{EL}$- concept descriptions. An *$\mathcal{EL}$-TBox* is a finite set of GCIs. Observe that TBoxes can be cyclic and allow for multiple inheritance. An *$\mathcal{EL}$-ABox* is a set of assertions of the form $C(a)$, or $r(a, b)$, where C is an $\mathcal{EL}$-concept description, $r \in N_R$, and $a, b \in N_I$. An *ontology* or *knowledge base* $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ consists of a TBox $\mathcal{T}$ and an ABox $\mathcal{A}$.

The semantics of $\mathcal{EL}$ is defined by means of interpretations $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consisting of a non-empty *domain* $\Delta^{\mathcal{I}}$ and an *interpretation function* $\cdot^{\mathcal{I}}$ that assigns binary relations on $\Delta^{\mathcal{I}}$ to role names, subsets of $\Delta^{\mathcal{I}}$ to concepts and elements of $\Delta^{\mathcal{I}}$ to individual names. The interpretation function $\cdot^{\mathcal{I}}$ is extended to concept descriptions in the usual way. For a more detailed description of the semantic of DLs see [3].

An interpretation $\mathcal{I}$ *satisfies* a concept inclusion $C \sqsubseteq D$, denoted as $\mathcal{I} \models C \sqsubseteq D$ if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$; it *satisfies* an assertion $C(a)$ (or $r(a, b)$), denoted as $\mathcal{I} \models C(a)$ ($\mathcal{I} \models r(a, b)$, resp.) if $a^{\mathcal{I}} \in C^{\mathcal{I}}$ ($(a^{\mathcal{I}}, b^{\mathcal{I}}) \in r^{\mathcal{I}}$, resp.). An interpretation $\mathcal{I}$ is a *model* of a knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ if it satisfies all GCIs in $\mathcal{T}$ and all assertions in $\mathcal{A}$.

We say that C is *subsumed* by D w.r.t. $\mathcal{T}$ (written $C \sqsubseteq_{\mathcal{T}} D$) if for every model $\mathcal{I}$ of $\mathcal{T}$ it holds that $\mathcal{I} \models C \sqsubseteq D$. The computation of the subsumption hierarchy of all named concepts in a TBox is called *classification*.

Finally, an individual $a \in N_I$ is an *instance* of a concept description C w.r.t. $\mathcal{K}$ (written $\mathcal{K} \models C(a)$) if $\mathcal{I} \models C(a)$ for all models $\mathcal{I}$ of $\mathcal{K}$. *ABox realization* is the task of computing, for each individual a in $\mathcal{A}$, the set of named concepts from $\mathcal{K}$ that have a as an instance and that are least (w.r.t. $\sqsubseteq$).

In this paper we are interested in computing generalizations by least common subsumers and most specific concepts, which we now formally define. Notice that our definition is general for any DL and not necessarily specific for $\mathcal{EL}$.

Definition 1 (least common subsumer). *Let $\mathcal{L}$ be a DL, $\mathcal{K}$ = ($\mathcal{T}$, $\mathcal{A}$) be a $\mathcal{L}$-KB. The* least common subsumer *(lcs) w.r.t. $\mathcal{T}$ of a collection of concepts $C_1, \ldots, C_n$ is the $\mathcal{L}$-concept description C such that*

1. *$C_i \sqsubseteq_\mathcal{T} C$ for all $1 \leq i \leq n$, and*
2. *for each $\mathcal{L}$-concept description D holds: if $C_i \sqsubseteq_\mathcal{T} D$ for all $1 \leq i \leq n$, then $C \sqsubseteq_\mathcal{T} D$.*

We will mostly consider the DL $\mathcal{EL}$ in this paper. Although defined as an n-ary operation, we will often write the lcs as a binary operation in the remainder of the paper for simplicity.

Definition 2 (most specific concept). *Let $\mathcal{L}$ be a DL, $\mathcal{K}$ = ($\mathcal{T}$, $\mathcal{A}$) be a $\mathcal{L}$-KB. The* most specific concept *(msc) w.r.t. $\mathcal{K}$ of an individual a from $\mathcal{A}$ is the $\mathcal{L}$-concept description C such that*

1. *$\mathcal{K} \models C(a)$, and*
2. *for each $\mathcal{L}$-concept description D holds: $\mathcal{K} \models D(a)$ implies $C \sqsubseteq_\mathcal{T} D$.*

Both inferences depend on the DL in use. For the DLs with conjunction as concept constructor the lcs and msc are, if exist, unique up to equivalence. Thus it is justified to speak of *the* lcs or *the* msc. Our computation methods for generalizations are based on the completion method, which we introduce in the following section.

3 Completion Algorithms for $\mathcal{EL}$

In principle, completion algorithms try to construct minimal models of the knowledge base. In case of classification algorithms such a model is constructed for the TBox and in case of ABox realization for the whole knowledge base. We describe the completion algorithm for ABox realization in $\mathcal{EL}$, originally described in [2], which can be easily restricted to obtain algorithms for classification. While the former is the basis for computing the role-depth bounded msc, the latter is used to obtain the role-depth bounded lcs.

For an $\mathcal{EL}$-KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ we want to test whether $\mathcal{K} \models D(a)$ holds. The completion algorithm first adds to $\mathcal{K}$ a concept name for the complex concept description D used in the instance check, i.e., $\mathcal{K} = (\mathcal{T} \cup \{A_q \equiv D\}, \mathcal{A})$, where A_q is a fresh concept name in $\mathcal{K}$. The instance checking algorithm for $\mathcal{EL}$ normalizes the knowledge base in two steps: first the ABox is transformed into a simple ABox. An ABox is a *simple ABox*, if it only contains concept names in concept assertions. An $\mathcal{EL}$-ABox $\mathcal{A}$ can be transformed into a simple ABox by first replacing each complex assertion $C(A)$ in $\mathcal{A}$ by $A(a)$ with a fresh name A and, second, introduce $A \equiv C$ in the TBox.

To describe the second normalization step, we need some notation. Let X be a concept description, a TBox, an ABox or a knowledge base. $\mathsf{CN}(X)$ denotes the set of all concept names and $\mathsf{RN}(X)$ denotes the set of all role names that appear in X. The *signature of X* (denoted $\mathsf{sig}(X)$) is then $\mathsf{CN}(X) \cup \mathsf{RN}(X)$. Now, an $\mathcal{EL}$-TBox $\mathcal{T}$ is in *normal form* if all concept axioms have one of the following forms, where $C_1, C_2 \in \mathsf{sig}(\mathcal{T})$ and $D \in \mathsf{sig}(\mathcal{T}) \cup \{\bot\}$:

$$C_1 \sqsubseteq D, \quad C_1 \sqcap C_2 \sqsubseteq D, \quad C_1 \sqsubseteq \exists r.C_2 \quad \text{or} \quad \exists r.C_1 \sqsubseteq D.$$

Any $\mathcal{EL}$-TBox can be transformed into normal form by introducing new concept names and by simply applying the normalization rules displayed in Figure 1 exhaustively.

$$\begin{array}{ll}
\textbf{NF1} & C \sqcap \hat{D} \sqsubseteq E \longrightarrow \{\, \hat{D} \sqsubseteq A, C \sqcap A \sqsubseteq E \,\} \\
\textbf{NF2} & \exists r.\hat{C} \sqsubseteq D \longrightarrow \{\, \hat{C} \sqsubseteq A, \exists r.A \sqsubseteq D \,\} \\
\textbf{NF3} & \hat{C} \sqsubseteq \hat{D} \longrightarrow \{\, \hat{C} \sqsubseteq A, A \sqsubseteq \hat{D} \,\} \\
\textbf{NF4} & B \sqsubseteq \exists r.\hat{C} \longrightarrow \{\, B \sqsubseteq \exists r.A, A \sqsubseteq \hat{C} \,\} \\
\textbf{NF5} & B \sqsubseteq C \sqcap D \longrightarrow \{\, B \sqsubseteq C, B \sqsubseteq D \,\}
\end{array}$$

where $\hat{C}, \hat{D} \notin \mathsf{CN}(\mathcal{T}) \cup \{\top\}$ and A is a new concept name.

Fig. 1. $\mathcal{EL}$ normalization rules

These rules replace the GCI on the left-hand side of the rules with the set of GCIs on the right-hand side.

Clearly, for a KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ the signature of $\mathcal{A}$ may be changed only during the first of the two normalization steps and the signature of $\mathcal{T}$ may be extended during both of the normalization steps. The normalization of the KB can be done in linear time.

The completion algorithm for instance checking is based on the one for classifying $\mathcal{EL}$-TBoxes introduced in [2]. Let $\mathcal{K}$ =($\mathcal{T}$, $\mathcal{A}$) be a normalized $\mathcal{EL}$-KB, i.e., with a simple ABox $\mathcal{A}$ and a TBox $\mathcal{T}$ in normal form. The completion algorithm works on four kinds of *completion sets*: $S(a), S(a,r), S(C)$ and $S(C,r)$ for each $a \in \mathsf{IN}(\mathcal{A})$, each $C \in \mathsf{CN}(\mathcal{K})$, and each $r \in \mathsf{RN}(\mathcal{K})$. The sets of the kind $S(a)$ and $S(a,r)$ contain individuals and concept names. The completion algorithm for classification uses only the latter two kinds of completion sets: $S(C)$ and $S(C,r)$, which contain only concept names from $\mathsf{CN}(\mathcal{K})$. Intuitively, the completion rules make implicit subsumption and instance relationships explicit in the following sense:

- $D \in S(C)$ implies that $C \sqsubseteq_{\mathcal{T}} D$,
- $D \in S(C,r)$ implies that $C \sqsubseteq_{\mathcal{T}} \exists r.D$.
- $D \in S(a)$ implies that a is an instance of D w.r.t. $\mathcal{K}$,
- $D \in S(a,r)$ implies that a is an instance of $\exists r.D$ w.r.t. $\mathcal{K}$.

$\mathsf{S}_{\mathcal{K}}$ denotes the set of all completion sets of a normalized $\mathcal{K}$. The completion sets are initialized for each $C \in \mathsf{CN}(\mathcal{K})$, each $r \in \mathsf{RN}(\mathcal{K})$, and each $a \in \mathsf{IN}(\mathcal{A})$ as follows:

- $S(C) := \{C, \top\}$
- $S(C,r) := \emptyset$
- $S(a) := \{C \in \mathsf{CN}(\mathcal{A}) \mid C(a) \text{ appears in } \mathcal{A}\} \cup \{\top\}$
- $S(a,r) := \{b \in \mathsf{IN}(\mathcal{A}) \mid r(a,b) \text{ appears in } \mathcal{A}\}$.

Then these sets are extended by applying the completion rules shown in Figure 2 until no more rule applies. In these rules C, C_1, C_2 and D are concept names and r is a role name, while X and Y can refer to concept or individual names in the algorithm for instance checking. In the algorithm for classification, X and Y refer to concept names. After the completion has terminated, the following relations hold between an individual a, a role r and named concepts A and B:

- subsumption relation between A and B from $\mathcal{K}$ holds iff $B \in S(A)$
- instance relation between a and B from $\mathcal{K}$ holds iff $B \in S(a)$,

which has been shown in [2].

CR1 If $C \in S(X)$, $C \sqsubseteq D \in \mathcal{T}$, and $D \notin S(X)$
then $S(X) := S(X) \cup \{D\}$

CR2 If $C_1, C_2 \in S(X)$, $C_1 \sqcap C_2 \sqsubseteq D \in \mathcal{T}$, and $D \notin S(X)$
then $S(X) := S(X) \cup \{D\}$

CR3 If $C \in S(X)$, $C \sqsubseteq \exists r.D \in \mathcal{T}$, and $D \notin S(X, r)$
then $S(X, r) := S(X, r) \cup \{D\}$

CR4 If $Y \in S(X, r)$, $C \in S(Y)$, $\exists r.C \sqsubseteq D \in \mathcal{T}$, and
$D \notin S(X)$ then $S(X) := S(X) \cup \{D\}$

Fig. 2. $\mathcal{EL}$ completion rules

To decide the initial query: $\mathcal{K} \models D(a)$, one has to test now, whether A_q appears in $S(a)$. In fact, instance queries for all individuals and all named concepts from the KB can be answered now; the completion algorithm does not only perform one instance check, but complete ABox realization. The completion algorithm for $\mathcal{EL}$ runs in polynomial time in size of the knowledge base.

4 Computing Role-Depth Bounded Generalizations

We employ the completion method now to compute first the role-depth bounded lcs and then the role-depth bounded msc in $\mathcal{EL}$.

4.1 Computing the Role-Depth Bounded LCS

As mentioned in the introduction, the lcs does not need to exist for cyclic TBoxes. Consider the TBox $\mathcal{T} = \{A \sqsubseteq \exists r.A \sqcap C,\ B \sqsubseteq \exists r.B \sqcap C\}$. The lcs of A and B is then

$$C \sqcap \exists r.(C \sqcap \exists r.(C \sqcap \exists r.(C \sqcap \cdots$$

and cannot be expressed by a finite concept description. To avoid such infinite nestings, we limit the role-depth of the concept description to be computed. The *role-depth* of a concept description C (denoted $rd(C)$) is the maximal number of nested quantifiers of C. Now we can define the lcs with limited role-depth.

Definition 3 (Role-depth bounded $\mathcal{L}$-lcs). *Let $\mathcal{T}$ be an $\mathcal{L}$-TBox and $C_1, \ldots, C_n$ $\mathcal{L}$-concept descriptions and $k \in \mathbb{N}$. Then the $\mathcal{L}$-concept description C is the* role-depth bounded $\mathcal{L}$-least common subsumer *of $C_1, \ldots, C_n$ w.r.t. $\mathcal{T}$ and role-depth k (written k-lcs$(C_1, \ldots, C_n)$) iff*

1. *$rd(C) \leq k$,*
2. *$C_i \sqsubseteq_{\mathcal{T}} C$ for all $1 \leq i \leq n$, and*
3. *for each $\mathcal{L}$-concept descriptions D with $rd(D) \leq k$ it holds that,*
   $C_i \sqsubseteq_{\mathcal{T}} D$ for all $1 \leq i \leq n$ implies $C \sqsubseteq_{\mathcal{T}} D$.

The computation algorithm for the role-depth bounded lcs w.r.t. general $\mathcal{EL}$-TBoxes, constructs the concept description from the set of completion sets. More precisely, it combines and intersects the completion sets in the same fashion as in the cross-product computation in the lcs algorithm for $\mathcal{EL}$-concept descriptions (without TBoxes) from [4]. The method we present here to compute the role-depth bounded lcs was described in [17].

However, the completion sets may contain concept names that were introduced during normalization. The returned lcs-concept description should only contain concept names that appear in the initial TBox, thus we need to "de-normalize" the concept descriptions obtained from the completion sets. However, the extension of the signature by normalization according to the normalization rules from Figure 1 does not affect subsumption tests for $\mathcal{EL}$-concept descriptions formulated w.r.t. the initial signature of $\mathcal{T}$. The following Lemma has been shown in [17].

Lemma 1. *Let $\mathcal{T}$ be an $\mathcal{EL}$-TBox and $\mathcal{T}'$ the TBox obtained from $\mathcal{T}$ by applying the $\mathcal{EL}$ normalization rules, C, D be $\mathcal{EL}$-concept descriptions with* $\mathsf{sig}(C) \subseteq \mathsf{sig}(\mathcal{T})$ *and* $\mathsf{sig}(D) \subseteq \mathsf{sig}(\mathcal{T}')$ *and D' be the concept description obtained by replacing all names* $A \in \mathsf{sig}(\mathcal{T}') \setminus \mathsf{sig}(\mathcal{T})$ *from D with $\top$. Then $C \sqsubseteq_{\mathcal{T}'} D$ iff $C \sqsubseteq_{\mathcal{T}} D'$.*

Lemma 1 guarantees that subsumption relations w.r.t. the normalized TBox $\mathcal{T}'$ between C and D, also hold w.r.t. the original TBox $\mathcal{T}$ for C and D', which is basically obtained from D by removing the names introduced by normalization, i.e., concept names from $\mathsf{sig}(\mathcal{T}') \setminus \mathsf{sig}(\mathcal{T})$.

We assume that the role-depth of each input concept of the lcs has a role-depth less or equal to k. This assumption is motivated by the applications of the lcs on the one hand and on the other by the simplicity of presentation, rather than a technical necessity. The algorithm for computing the role-depth bounded lcs of two $\mathcal{EL}$-concept descriptions is depicted in Algorithm 1.

The procedure k-lcs first adds concept definitions for the input concept descriptions to (a copy of) the TBox and transforms this TBox into the normalized TBox $\mathcal{T}'$. Next, it calls the procedure apply-completion-rules, which applies the $\mathcal{EL}$ completion rules exhaustively to the TBox $\mathcal{T}'$, and stores the obtained set of completion sets in S. Then it calls the function k-lcs-r with the concept names A and B for the input concepts, the set of completion sets S, and the role-depth limit k. The result is then de-normalized and returned (lines 4 to 6). More precisely, in case a complex concept description is returned from k-lcs-r, the procedure remove-normalization-names removes concept names that were added during the normalization of the TBox.

The function k-lcs-r gets a pair of concept names, a set of completion sets and a natural number as inputs. First, it tests whether one of the input concepts subsumes the other w.r.t. $\mathcal{T}'$. In that case the name of the subsuming concept is returned. Otherwise the set of concept names that appear in the completion sets of both input concepts is stored in common-names (line 4).[2] In case the role-depth bound is reached ($k = 0$), the conjunction of the elements in common-names is returned. Otherwise, the elements in common-names are conjoined with a conjunction over all roles $r \in \mathsf{RN}(\mathcal{T})$, where for each r and each element of the cross-product over the r-successors of the current A and B a recursive call to k-lcs-r is made with the role-depth bound reduced by 1 (line 6). This conjunction is then returned to k-lcs.

[2] Note, that the intersection $S(A) \cap S(B)$ is never empty, since both sets contain $\top$.

Algorithm 1. Computation of a role-depth bounded $\mathcal{EL}$-lcs.

```
Procedure k-lcs $(C, D, \mathcal{T}, k)$
Input: $C, D$: $\mathcal{EL}$-concept descriptions; $\mathcal{T}$: $\mathcal{EL}$-TBox; $k$: natural number
Output: $k\text{-}lcs(C, D)$: role-depth bounded $\mathcal{EL}$-lcs of $C$ and $D$ w.r.t $\mathcal{T}$ and $k$.
1: $\mathcal{T}'$ := normalize($\mathcal{T} \cup \{A \equiv C, B \equiv D\}$)
2: $\mathsf{S}_{\mathcal{T}'}$ := apply-completion-rules($\mathcal{T}'$)
3: $L$ := k-lcs-r $(A, B, \mathsf{S}_{\mathcal{T}'}, k)$
4: if $L = A$ then return $C$
5: else if $L = B$ then return $D$
6: else return remove-normalization-names($L$)
7: end if

Procedure k-lcs-r $(A, B, \mathsf{S}, k)$
Input: $A, B$: concept names; $\mathsf{S}$: set of completion sets; $k$: natural number
Output: $k\text{-}lcs(A, B)$: role-depth bounded $\mathcal{EL}$-lcs of $A$ and $B$ w.r.t $\mathcal{T}$ and $k$.
1: if $B \in S(A)$ then return $B$
2: else if $A \in S(B)$ then return $A$
3: end if
4: common-names := $S(A) \cap S(B)$
5: if $k = 0$ then return $\bigsqcap_{P \in \text{common-names}} P$
6: else return $\bigsqcap_{P \in \text{common-names}} P \sqcap$
       $\bigsqcap_{r \in \mathsf{RN}(\mathcal{T})} \Big( \bigsqcap_{(E,F) \in S(A,r) \times S(B,r)} \exists r.\ \text{k-lcs-r}\ (E, F, \mathsf{S}, k-1)\Big)$
7: end if
```

For $L = \text{k-lcs}(C, D, \mathcal{T}, k)$ it holds by construction that $rd(L) \leq k$.[3] We now show that the result of the function k-lcs is a common subsumer of the input concept descriptions. It was shown in [17] that all conditions of Definition 3 are fulfilled for k-lcs$(C, D, \mathcal{T}, k)$.

Theorem 1. *Let C and D be $\mathcal{EL}$-concept descriptions, $\mathcal{T}$ an $\mathcal{EL}$-TBox, $k \in \mathbb{N}$, then* k-lcs$(C, D, \mathcal{T}, k) \equiv k\text{-}lcs(C, D)$.

For cases where k-lcs returns a concept description with role-depth of less than k we conjecture that it is the exact lcs.

The complexity of the overall method is exponential. However, if a compact representation of the lcs with structure sharing is used, the lcs-concept descriptions can be represented polynomially.

If a $k\text{-}lcs$ is too general and a bigger role depth of the $k\text{-}lcs$ is desired, the completion of the TBox does not have to be redone for a second computation. The completion sets can simply be "traversed" further.

4.2 Computing the Role-Depth Bounded MSC

The msc was first investigated for $\mathcal{EL}$-concept descriptions and w.r.t. unfoldable TBoxes and possibly cyclic ABoxes in [12]. Similar to the lcs, the msc does not need to exist,

[3] Recall our assumption: the role-depth of each input concept is less or equal to k.

since cyclic structures cannot be expressed by $\mathcal{EL}$-concept descriptions. Now we can define the msc with limited role-depth.

Definition 4 (role-depth bounded $\mathcal{L}$-msc). *Let $\mathcal{K}$ =($\mathcal{T}$, $\mathcal{A}$) be a $\mathcal{L}$-KB and a an individual in $\mathcal{A}$ and $k \in \mathbb{N}$. Then the $\mathcal{L}$-concept description C is the* role-depth bounded $\mathcal{EL}$-most specific concept *of a w.r.t. $\mathcal{K}$ and role-depth k (written k-$msc_{\mathcal{K}}(a)$) iff*

1. *$rd(C) \leq k$,*
2. *$\mathcal{K} \models C(a)$, and*
3. *for each $\mathcal{EL}$-concept description D with $rd(D) \leq k$ holds: $\mathcal{K} \models D(a)$ implies $C \sqsubseteq_{\mathcal{T}} D$.*

In case the exact msc has a role-depth less than k the role-depth bounded msc is the exact msc.

Again, we construct the msc by traversing the completion sets to "collect" the msc. More precisely, the set of completion sets encodes a graph structure, where the sets $S(X)$ are the nodes and the sets $S(X, r)$ encode the edges. Traversing this graph structure, one can construct an $\mathcal{EL}$-concept. To obtain a finite concept in the presence of cyclic ABoxes or TBoxes one has to limit the role-depth of the concept to be obtained.

Definition 5 (traversal concept). *Let $\mathcal{K}$ be an $\mathcal{EL}$-KB, $\mathcal{K}''$ be its normalized form, $\mathsf{S}_{\mathcal{K}}$ the completion set obtained from $\mathcal{K}$ and $k \in \mathbb{N}$. Then the* traversal concept of a named concept *A (denoted k-$\mathsf{C}_{\mathsf{S}_{\mathcal{K}}}(A)$) with* $\mathsf{sig}(A) \subseteq \mathsf{sig}(\mathcal{K}'')$ *is the concept obtained from executing the procedure call* traversal-concept-c*(A, $\mathsf{S}_{\mathcal{K}}$, k) shown in Algorithm* [2].

The traversal concept of an individual *a (denoted k-$\mathsf{C}_{\mathsf{S}_{\mathcal{K}}}(a)$) with $a \subseteq$* $\mathsf{sig}(\mathcal{K})$ *is the concept description obtained from executing the procedure call* traversal-concept-i*(a, $\mathsf{S}_{\mathcal{K}}$, k) shown in Algorithm* [2].

The idea is that the traversal concept of an individual yields its msc. However, the traversal concept contains names that were introduced during normalization. The returned msc should be formulated w.r.t. the signature of the original KB, thus the normalization names need to be removed or replaced.

Lemma 2. *Let $\mathcal{K}$ be an $\mathcal{EL}$-KB, $\mathcal{K}''$ its normalized version, $\mathsf{S}_{\mathcal{K}}$ be the set of completion sets obtained for $\mathcal{K}$, $k \in \mathbb{N}$ a natural number and $a \in \mathsf{IN}(\mathcal{K})$. Furthermore let $C = k$-$\mathsf{C}_{\mathsf{S}_{\mathcal{K}}}(a)$ and $\widehat{C}$ be obtained from C by removing the normalization names. Then*

$$\mathcal{K}'' \models C(a) \text{ iff } \mathcal{K} \models \widehat{C}(a).$$

This lemma guarantees that removing the normalization names from the traversal concept preserves the instance relationships. Intuitively, this lemma holds since the construction of the traversal concept conjoins exhaustively all named subsumers and all subsuming existential restrictions to a normalization name up to the role-depth bound. Thus removing the normalization name does not change the extension of the conjunction. The proof can be found in [18]. We are now ready to devise a computation algorithm for the role-depth bounded msc: procedure k-msc as displayed in Algorithm [2].

The procedure k-msc has an individual a from a knowledge base $\mathcal{K}$, the knowledge base $\mathcal{K}$ itself and number k for the role depth-bound as parameter. It first performs the

Algorithm 2. Computation of a role-depth bounded $\mathcal{EL}$-msc.

```
Procedure k-msc $(a, \mathcal{K}, k)$
Input: $a$: individual from $\mathcal{K}$; $\mathcal{K}$ =($\mathcal{T}$, $\mathcal{A}$) an $\mathcal{EL}$-KB; $k \in \mathbb{N}$
Output: role-depth bounded $\mathcal{EL}$-msc of $a$ w.r.t. $\mathcal{K}$ and $k$.
1: $(\mathcal{T}', \mathcal{A}')$ := simplify-ABox($\mathcal{T}$, $\mathcal{A}$)
2: $\mathcal{K}''$ := (normalize($\mathcal{T}'$), $\mathcal{A}'$)
3: $\mathsf{S}_\mathcal{K}$ := apply-completion-rules($\mathcal{K}$)
4: return Remove-normalization-names ( traversal-concept-i$(a, \mathsf{S}_\mathcal{K}, k)$)

Procedure traversal-concept-i $(a, \mathsf{S}, k)$
Input: $a$: individual name from $\mathcal{K}$; S: set of completion sets; $k \in \mathbb{N}$
Output: role-depth traversal concept (w.r.t. $\mathcal{K}$) and $k$.
1: if $k = 0$ then return $\bigsqcap_{A \in S(a)} A$
2: else return $\bigsqcap_{A \in \mathsf{S}(a)} A \sqcap$
       $\bigsqcap_{r \in \mathsf{RN}(\mathcal{K}'')} \bigsqcap_{A \in \mathsf{CN}(\mathcal{K}'') \cap S(a,r)} \exists r.$ traversal-concept-c $(A, \mathsf{S}, k-1) \sqcap$
       $\bigsqcap_{r \in \mathsf{RN}(\mathcal{K}'')} \bigsqcap_{b \in \mathsf{IN}(\mathcal{K}'') \cap S(a,r)} \exists r.$ traversal-concept-i $(b, \mathsf{S}, k-1)$
3: end if

Procedure traversal-concept-c $(A, \mathsf{S}, k)$
Input: $A$: concept name from $\mathcal{K}''$; S: set of completion sets; $k \in \mathbb{N}$
Output: role-depth bounded traversal concept.
1: if $k = 0$ then return $\bigsqcap_{B \in S(A)} B$
2: else return $\bigsqcap_{B \in S(A)} B \sqcap \bigsqcap_{r \in \mathsf{RN}(\mathcal{K}'')} \bigsqcap_{B \in S(A,r)} \exists r.$traversal-concept-c $(B, \mathsf{S}, k-1)$
3: end if
```

two normalization steps on $\mathcal{K}$, then applies the completion rules from Figure 2 to the normalized KB $\mathcal{K}''$ and stores the set of completion sets in $\mathsf{S}_\mathcal{K}$. Afterwards it computes the traversal-concept of a from $\mathsf{S}_\mathcal{K}$ w.r.t. role-depth bound k. In a post-processing step it applies Remove-normalization-names to the traversal concept.

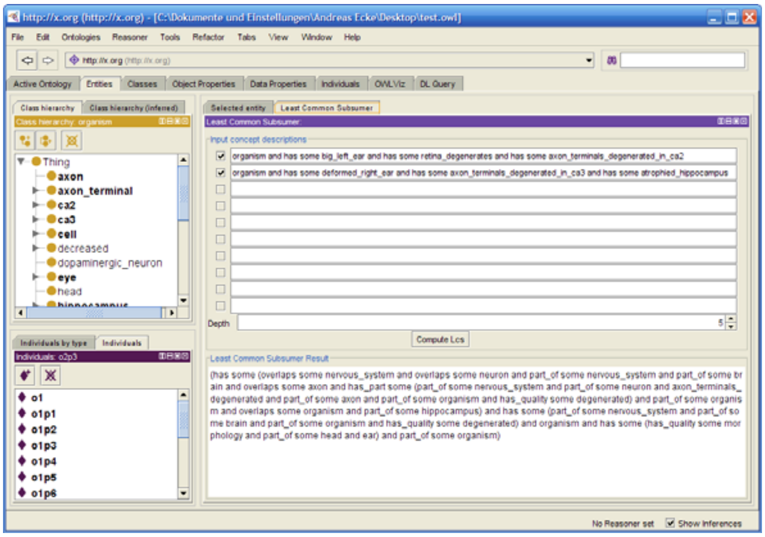
Obviously, the concept description returned from the procedure k-msc has a role-depth less or equal to k. The other conditions of Definition 4 are fulfilled as well, which has been shown in [18] yielding the correctness of the overall procedure.

Theorem 2. *Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be an $\mathcal{EL}$-KB and a an individual in $\mathcal{A}$ and $k \in \mathbb{N}$. Then* k-msc$(a, \mathcal{K}, k) \equiv k\text{-}msc_\mathcal{K}(a)$.

The k-msc can grow exponential in the size of the knowledge base.

5 Implementation of GEL

The completion algorithm for classifying $\mathcal{EL}$ TBoxes was first implemented in the CEL reasoner [5]. We used its successor system JCEL [16] as a starting point for our implementation for the computation of the role-depth bounded lcs and msc. The implementation was done in Java and provides a simple GUI for the ontology editor PROTÉGÉ as can be seen in the screen-shot in Figure 3.



```
organism and has some big_left_ear and has some retina_degenerates and has some axon_terminals_degenerated_in_ca2
organism and has some deformed_right_ear and has some axon_terminals_degenerated_in_ca3 and has some atrophied_hippocampus
```

```
(has some (overlaps some nervous_system and overlaps some neuron and part_of some nervous_system and part_of some br
ain and overlaps some axon and has_part some (part_of some nervous_system and part_of some neuron and axon_terminals_
degenerated and part_of some axon and part_of some organism and has_quality some degenerated) and part_of some organis
m and overlaps some organism and part_of some hippocampus) and has some (part_of some nervous_system and part_of so
me brain and part_of some organism and has_quality some degenerated) and organism and has some (has_quality some mor
phology and part_of some head and ear) and part_of some organism)
```

Fig. 3. LCS plugin

Our implementation of the methods presented here accesses the internal data structures of JCEL directly, providing a full integration of GEL into JCEL. The reasoning methods in GEL are in this first version realized in a naive way and are still in need of optimizations in order to handle the large knowledge bases that can be encountered in practice.

The concept descriptions returned by the lcs and the msc can grow exponentially in the worst case. On top of that, the returned concept descriptions are quite redundant in our current implementation, which might be acceptable if used as an input for a similarity measure, but surely not if presented to a human reader. It is future work to investigate methods for minimal rewritings of concept descriptions w.r.t. a general $\mathcal{EL}$ knowledge base in order to be able to present redundancy-free concept descriptions. Our tool will be made available as a plug-in for the ontology editor PROTÉGÉ and an API for the k-limited lcs and -msc is planned. The former system sonic [24] implemented the lcs and msc as well, but allowed only for acyclic, unfoldable TBoxes.

6 Conclusions

In this paper we have presented a practical method for computing the role-depth bounded lcs and the role-depth bounded msc of $\mathcal{EL}$-concepts w.r.t. a general TBox. We have argued that such generalization inferences are useful for ontology-based applications in many ways. Our approach for computing (approximations of) these inferences is based on the completion sets that are computed during classification of a TBox or realization of an ABox. Thus, any of the available implementations of the $\mathcal{EL}$ completion algorithm can be easily extended to an implementation of the two generalization inferences considered here. The same idea can be adapted for the computation of generalizations in the probabilistic DL Prob-$\mathcal{EL}_c^{01}$ [17,18].

These theoretical results complete the (approximative) bottom-up approach for general $\mathcal{EL}$- (and Prob-$\mathcal{EL}_c^{01}$-) KBs. Continuing on the theoretical side, we want to investigate the bottom-up constructions (i.e. lcs and msc computations) in more expressive members of the $\mathcal{EL}$-family. We want to extend the approximative methods to $\mathcal{EL}$++, which extends $\mathcal{EL}$, for example, by transitive roles and role hierarchies. Such an extension would enable generalization reasoning services for the OWL 2 EL profile. Another interesting extension is to allow for more expressive means for probabilities.

Although a non-redundant representation of the concept descriptions obtained by the approximative lcs and msc is desirable when presented to a human reader, it is not clear whether a minimal representation of the obtained concept descriptions is favorable in every case. It might depend on the similarity measures employed whether a redundant representation of a concept is preferable over a compact one.

On the practical side, our future work will include evaluations of the usefulness of the offered reasoning services for biomedical applications and the development and testing of optimizations regarding the performance of the implementation.

Acknowledgments. We would like to thank Andreas Ecke and Julian Mendez for their implementation effort.

References

1. Baader, F.: Least common subsumers and most specific concepts in a description logic with existential restrictions and terminological cycles. In: Gottlob, G., Walsh, T. (eds.) Proc. of the 18th Int. Joint Conf. on Artificial Intelligence (IJCAI 2003), pp. 325–330. Morgan Kaufmann, San Francisco (2003)
2. Baader, F., Brandt, S., Lutz, C.: Pushing the $\mathcal{EL}$ envelope. In: Proc. of the 19th Int. Joint Conf. on Artificial Intelligence (IJCAI 2005), Edinburgh, UK. Morgan-Kaufmann Publishers, San Francisco (2005)
3. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P. (eds.): The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press, Cambridge (2003)
4. Baader, F., Küsters, R., Molitor, R.: Computing least common subsumers in description logics with existential restrictions. In: Dean, T. (ed.) Proc. of the 16th Int. Joint Conf. on Artificial Intelligence (IJCAI 1999), Stockholm, Sweden, pp. 96–101. Morgan Kaufmann, Los Altos (1999)
5. 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), CEL download page `http://lat.inf.tu-dresden.de/systems/cel/`
6. Borgida, A., Walsh, T., Hirsh, H.: Towards measuring similarity in description logics. In: Proc. of the 2005 Description Logic Workshop (DL 2005). CEUR Workshop Proceedings, vol. 147 (2005)
7. Cohen, W.W., Borgida, A., Hirsh, H.: Computing least common subsumers in description logics. In: Swartout, W. (ed.) Proc. of the 10th Nat. Conf. on Artificial Intelligence (AAAI 1992), San Jose, CA, pp. 754–760. AAAI Press/The MIT Press (1992)

8. Colucci, S., Di Sciascio, E., Donini, F.M., Tinelli, E.: Partial and informative common subsumers in description logics. In: Proc. of 18th European Conference on Artificial Intelligence (ECAI 2008). IOS Press, Amsterdam (2008)
9. Consortium, T.G.O.: Gene Ontology: Tool for the unification of biology. Nature Genetics 25, 25–29 (2000)
10. d'Amato, C., Fanizzi, N., Esposito, F.: A semantic similarity measure for expressive description logics. In: Proc. of Convegno Italiano di Logica Computazionale, CILC 2005 (2005)
11. Keet, M.C., Roos, M., Marshall, M.S.: A survey of requirements for automated reasoning services for bio-ontologies in OWL. In: Proceedings of Third international Workshop OWL: Experiences and Directions, OWLED 2007 (2007)
12. Küsters, R., Molitor, R.: Approximating most specific concepts in description logics with existential restrictions. AI Communications 15(1), 47–59 (2002)
13. Lord, P.W., Stevens, R.D., Brass, A., Goble, C.A.: Investigating semantic similarity measures across the gene ontology: The relationship between sequence and annotation. Bioinformatics 19(10), 1275–1283 (2003)
14. Lutz, C., Piro, R., Wolter, F.: Enriching el-concepts with greatest fixpoints. In: Proc. of the 19th European Conf. on Artificial Intelligence (ECAI 2010). IOS Press, Amsterdam (2010)
15. Mantay, T., Möller, R.: Content-based information retrieval by computation of least common subsumers in a probabilistic description logic. In: Proc. International Workshop on Intelligent Information Integration, ECAI 1998, Brighton, UK, August 23-28 (1998)
16. Mendez, J., Suntisrivaraporn, B.: Reintroducing CEL as an OWL 2 EL reasoner. In: Cuenca Grau, B., Horrocks, I., Motik, B., Sattler, U. (eds.) Proc. of the 2008 Description Logic Workshop (DL 2009). CEUR-WS, vol. 477 (2009)
17. Peñaloza, R., Turhan, A.-Y.: Role-depth bounded least common subsumers by completion for $\mathcal{EL}$- and Prob-$\mathcal{EL}$-TBoxes. In: Haarslev, V., Toman, D., Weddell, G. (eds.) Proc. of the 2010 Description Logic Workshop, DL 2010 (2010)
18. Peñaloza, R., Turhan, A.-Y.: Towards approximative most specific concepts by completion for $\mathcal{EL}^{01}$ with subjective probabilities. In: Lukasiewicz, T., Peñaloza, R., Turhan, A.-Y. (eds.) Proceedings of the First International Workshop on Uncertainty in Description Logics, UniDL 2010 (2010)
19. Pesquita, C., Faria, D., Falco, A.O., Lord, P., Couto, F.M.: Semantic similarity in biomedical ontologies. PLoS Comput. Biol. 5 (2009)
20. Resnik, P.: Using information content to evaluate semantic similarity in a taxonomy. In: Proc. of the 14th Int. Joint Conf. on Artificial Intelligence (IJCAI 1995), pp. 448–453 (1995)
21. Spackman, K.: Managing clinical terminology hierarchies using algorithmic calculation of subsumption: Experience with snomed-rt. Journal of the American Medical Informatics Assoc. (2000); Fall Symposium Special Issue
22. Springer, T., Turhan, A.-Y.: Employing description logics in ambient intelligence for modeling and reasoning about complex situations. Journal of Ambient Intelligence and Smart Environments 1(3), 235–259 (2009)
23. Turhan, A.-Y.: On the Computation of Common Subsumers in Description Logics. PhD thesis, TU Dresden, Institute for Theoretical Computer Science (2007)
24. Turhan, A.-Y., Kissig, C.: SONIC — non-standard inferences go OILED. In: Basin, D., Rusinowitch, M. (eds.) IJCAR 2004. LNCS (LNAI), vol. 3097, pp. 321–325. Springer, Heidelberg (2004), SONIC is available from `http://wwwtcs.inf.tu-dresden.de/~sonic/`
25. W3C OWL Working Group. OWL 2 web ontology language document overview. W3C Recommendation (October 27, 2009), `http://www.w3.org/TR/2009/REC-owl2-overview-20091027/`

Author Index

Abel, Fabian II-375
Adolf, Bob II-31
Alani, Harith II-405
Allocca, Carlo I-352
al-Saffar, Sinan II-31
Álvaro, Guillermo II-154
Andrews, Pierre II-476
Angeletou, Sofia II-405
Anicic, Darko II-270
Anyanwu, Kemafor II-46, II-123
Arenas, Marcelo II-1
Arndt, Natanael I-200
Auer, Sören I-185, I-200, II-456
Au Yeung, Ching-man I-32

Babitski, Grigori II-183
Bae, Hyunjoo II-446
Barzdins, Guntis II-441
Beckmann, Dirk II-198
Benz, Dominik II-360
Bergweiler, Simon II-183
Bicer, Veli I-47
Blansché, Alexandre II-421
Blum, Daniel II-16
Borillo Domenech, Ricardo I-17
Bouayad-Agha, Nadjet I-230
Brennan, Rob I-336
Breslin, John G. II-390
Bühmann, Lorenz I-63
Buil-Aranda, Carlos II-1

Cai, Zhonglun I-32
Calbimonte, Jean-Paul II-300
Cano, A. Elizabeth II-451
Carbone, Francesco II-154
Casamayor, Gerard I-230
Castagnone, Michelangelo II-154
Chen, Jianfeng I-290
Chen, Ling II-123
Cimiano, Philipp I-245
Cohen, Sara II-16
Constantopoulos, Panos II-77
Contreras, Jesús II-154
Corcho, Oscar II-1, II-300, II-330
Córdoba, Carmen II-154
Correndo, Gianluca II-139
Curé, Olivier I-367

Dadzie, Aba-Sah I-93, II-451
Damova, Mariana I-80
d'Aquin, Mathieu II-461
David, Catalin II-471
Deligiannakis, Antonios II-77
De Roure, David II-300
Dietze, Stefan II-436
Díez, Fernando I-230
Domingue, John II-436
Dritsou, Vicky II-77

Ehrlich, Christian II-456
Elahi, Salman II-492
Elbassuoni, Shady II-62
Ermilov, Timofey I-185, I-200

Farazi, Feroz II-169
Farfeleder, Stefan II-212
Fernandes, Alvaro A.A. II-300
Ferrara, Alfio II-108
Fleischhacker, Daniel II-508
Frazer, Alex II-300
Frischmuth, Philipp I-200
Frosterus, Matias II-243
Fu, Bo I-336

Galpin, Ixent II-300
Gao, Qi II-375
Gao, Sidan II-123
Gao, Zhiqiang II-497
García-Castro, Raúl II-300
Gibbins, Nicholas I-32
Gibbins, Nick II-139
Ginev, Deyan II-471
Giunchiglia, Fausto II-169
Glimm, Birte I-382
Gómez-Pérez, Asunción II-300
Gómez-Pérez, José Manuel II-154
Goodman, Carl II-466
Goodman, Eric L. II-31
Gossen, Anna I-47

Gray, Alasdair J.G. II-300
Grebner, Olaf II-183
Grimm, Stephan I-260, I-397, II-345
Gutierrez, Claudio II-93

Haak, Steffen II-345
Haase, Peter II-481
Haglin, David II-31
Hall, Wendy I-32
Harris, Steve II-139
Harth, Andreas I-170
Hartig, Olaf I-154
Hartmann, Melanie II-451
Hayes, Conor II-390
Heino, Norman I-185
Hellmann, Sebastian II-456
Herzig, Daniel M. II-518
Hirota, Takeru I-305
Hitzler, Pascal I-80
Horrocks, Ian I-382
Hose, Katja II-481
Hotho, Andreas II-360
Houben, Geert-Jan II-375
Hu, Wei I-290
Huang, Zhisheng II-497
Hurtado, Carlos II-93
Hyvönen, Eero I-215, II-243, II-255

Ivanyukovich, Alexander II-169

Jain, Prateek I-80
Ji, Qiu II-497
Jimenez, Edward II-31
Jovanovic, Jelena I-108
Jung, Yuchul II-446

Kanshin, Sergey II-476
Karpathiotakis, Manos II-300
Keet, C. Maria I-321
Kim, Hwa-Sung II-446
Kim, HyeongSik II-46
Kinsella, Sheila II-390
Klebeck, Alex II-456
Kohlhase, Andrea II-471
Kohlhase, Michael II-471
Kollia, Ilianna I-382, II-513
Körner, Christian II-360
Köster, Frank II-198
Kotidis, Yannis II-77
Koubarakis, Manolis II-300
Koutsomitropoulos, Dimitrios A. I-17
Kozaki, Kouji I-305
Krall, Andreas II-212
Kyzirakos, Kostis II-300

Ladwig, Günter I-139
Laitio, Joonas II-243
Lamolle, Myriam I-367
Lange, Christoph II-471
Laublet, Philippe I-108
Laurenne, Nina II-255
Le Duc, Chan I-367
Lee, Deirdre II-227
Lehmann, Jens I-63
Leidinger, Lucas II-285
Liu, Dong II-436
López Hernández, Sergio I-230
Loutas, Nikolaos II-227

Maali, Fadi II-227
Maltese, Vincenzo II-169
Manolis, Nikos I-1
Martinez, Kirk II-300
Matican, Bogdan II-471
McCrae, John I-245
Mear, David II-466
Mehdi, Anees I-397
Mirea, Stefan II-471
Mizell, David II-31
Mizoguchi, Riichiro I-305
Montanelli, Stefano II-108
Moser, Thomas II-212
Motta, Enrico II-461

Napoli, Amedeo II-421
Niepert, Mathias I-124
Niu, Xing I-275
Noessner, Jan II-108
Noyer, Ulf II-198

Oberle, Daniel II-183
Omoronyia, Inah II-212
O'Sullivan, Declan I-336

Paar, Alexander II-315
Page, Kevin II-300
Pane, Juan II-476
Park, Yoo-mi II-446
Paton, Norman W. II-300
Paulheim, Heiko II-183

Pedrinaci, Carlos II-436
Peñaloza, Rafael I-410
Penela, Víctor II-154
Peristeras, Vassilios II-227
Petrelli, Daniela I-93
Probst, Florian II-183

Qi, Guilin I-275
Qu, Yuzhong I-290

Ramanath, Maya II-62
Ravindra, Padmashree II-46
Rowe, Matthew I-93, II-405
Rudolph, Sebastian I-397
Ruiz, Carlos II-154

Sadler, Jason II-300
Salvadores, Manuel I-32, II-139
Saquicela, Víctor II-330
Schenkel, Ralf II-481
Schmidt, Michael II-481
Schneider, Michael II-487
Schwarte, Andreas II-481
Segers, Roxane II-503
Shadbolt, Nigel I-32, II-139
Sheth, Amit P. I-80
Shi, Lian II-421
Simperl, Elena II-466
Singh, Priyanka I-32
Sinkkilä, Reetta I-215
Siorpaes, Katharina II-466
Solomou, Georgia D. I-17
Speiser, Sebastian I-170
Spohr, Dennis I-245
Stålhane, Tor II-212
Stankovic, Milan I-108
Stojanovic, Ljiljana II-270
Stojanovic, Nenad II-270
Strohmaier, Markus II-360
Stuckenschmidt, Heiner II-108
Studer, Rudi II-270
Stumme, Gerd II-360
Suominen, Osma I-215

Tao, Ke II-375
Tarabanis, Konstantinos II-227
Taylor, Kerry II-285
Thaler, Stefan II-466
Toussaint, Yannick II-421
Tramp, Sebastian I-185, I-200
Tran, Thanh I-47, I-139
Tuominen, Jouni II-255
Turhan, Anni-Yasmin I-410
Tzitzikas, Yannis I-1

Vaisman, Alejandro II-93
Vasquez, Reymonrod G. I-80
Verma, Kunal I-80
Vilches-Blazquez, Luis.M. II-330
Völker, Johanna I-124
Vrandečić, Denny II-315

Wang, Haofen I-275
Wang, Mengjiao II-390
Wang, Xiaowei I-32
Wanner, Leo I-230
Weikum, Gerhard II-62
Wissmann, Jens I-260
Wu, Gang I-275

Xu, Yongchun II-270

Yang, Yang I-32
Yao, Jiadi I-32
Yeh, Peter Z. I-80
Yoo, HyunKyung II-446
Yu, Hong Qing II-436
Yu, Yong I-275

Zablith, Fouad II-461
Zaihrayeu, Ilya II-476
Zareian, Amir I-32
Zhang, Hang I-290
Zholudev, Vyacheslav II-471
Zojer, Herbert II-212
Zviedris, Martins II-441